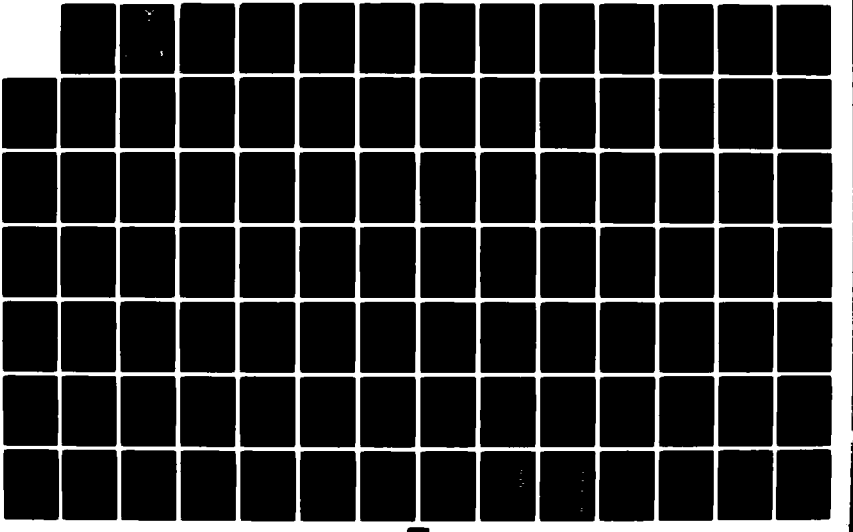
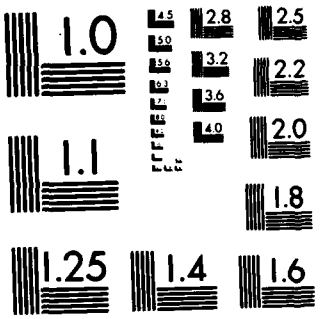
AD-A124 758 A STUDY OF THE SOFTWARE MAINTENANCE PROCESS OF AIR FORCE WEAPON SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING 1/2

UNCLASSIFIED J P JOYCE DEC 82 AFIT/GCS/MA/82D-5 F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

A STUDY OF THE SOFTWARE MAINTENANCE PROCESS OF AIR FORCE WEAPON SYSTEMS

THESIS

AFIT/GCS/MA/82D-5

James P. Joyce
CAPT USAF

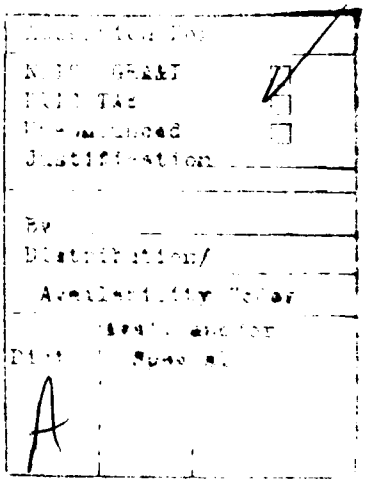
Approved for Public Release; Distribution Unlimited

A STUDY OF THE
SOFTWARE MAINTENANCE PROCESS
OF
AIR FORCE WEAPON SYSTEMS

THESIS

Presented to the Faculty of the School of Civil Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Masters of Science

BY
JAMES P. JOYCE
CAPT USAF
GRADUATE COMPUTER SCIENCE
DECEMBER 1982



Approved for Public Release; Distribution Unlimited

Preface

The majority of recent software engineering advancements have been directed towards improving the methods of developing software. Relatively little work has been devoted to the study of methods to maintain software once it has been developed. Yet, the rising cost of software maintenance is causing a change in emphasis. Indeed, the cost of supporting software within Air Force weapon systems is becoming a major concern. The purpose of this study was to investigate the software maintenance process of Air Force weapon systems in order to identify methods to reduce software life cycle costs. In accomplishing this objective, this research provides a profile of maintenance management and programming methods in use at selected Air Force systems. By describing and analyzing the attributes of this specific software application, this work provides a valuable contribution to the evolving field of software maintenance engineering and outlines directions for further research and experimentation.

The findings of this research are based on information obtained from interviews of many knowledgeable personnel within several Air Force organizations. I thank these individuals for the time, patience, and effort extended in answering my many inquiries. I would also like to thank Lt Col Mueller, Air Force Test and Evaluation Center (AFTEC/LG5) and his staff for sponsoring this thesis and

providing extensive expertise and assistance during this research effort. Lastly, I extend sincere thanks to Professor Daniel E. Reynolds, Assistant Professor of Computing Sciences at the Air Force Institute of Technology for his guidance and support in completion of this study. As thesis advisor, Professor Reynolds was of invaluable assistance in providing research directions and in preparing this report.

The interaction with these people coupled with the challenges presented by the research subject have made this thesis effort a stimulating, demanding and rewarding personal experience.

JAMES P. JOYCE

Contents

List of Figures

List of Tables

Abstract

The increasing cost of software maintenance is becoming a critical concern. This master thesis profiles Air Force software maintenance activities and provides recommendations for improving management and programmer efforts. The software maintenance activities of thirteen Air Force weapon systems were investigated through structured interviews of key management and programmer personnel. Analysis of interview responses resulted in the identification of four factors which effect the level of maintenance effort: system age, personnel experience, documentation and code quality, and level of user enhancements. Interview responses also identified three major management policy issues which cause the Air Force to make excess resource commitments to software maintenance functions:

1) Personnel experience
2) Software development and life cycle planning, and
3) Software development/maintenance standards.

Management policy recommendations were developed to reduce the impact of these conditions on Air Force weapon system software maintenance activities.

I. Introduction

Computer Technology The Software Investment

The cost of computer software is increasingly becoming a critical issue. The total United States expenditure for computer programming in 1977 is estimated to have exceeded $50 billion and probably approached $100 billion (Ref 44:1060). The investment in software is predicted to grow exponentially. By 1995, the yearly software investment is likely to be over $650 billion or 21 percent of the United States GNP. Growth of this measure justifies concern.

A breakdown of total data processing costs shows that software costs have steadily risen from less than 20 percent in 1955 to more than 80 percent today. Figure 1-1 charts the reduction of hardware processing costs and the increase in cost of programmer personnel. From 1955 to 1975, advances in hardware technology have increased the potential processing speed over 800,000 percent and improved the performance/cost ratio 1000 percent. In the same period of time, advances in software tools and methods have improved programmer productivity six times (Ref 9:88). In order to keep pace with the software growth demands, the population of programmers has grown to over 300,000 within 25 years.

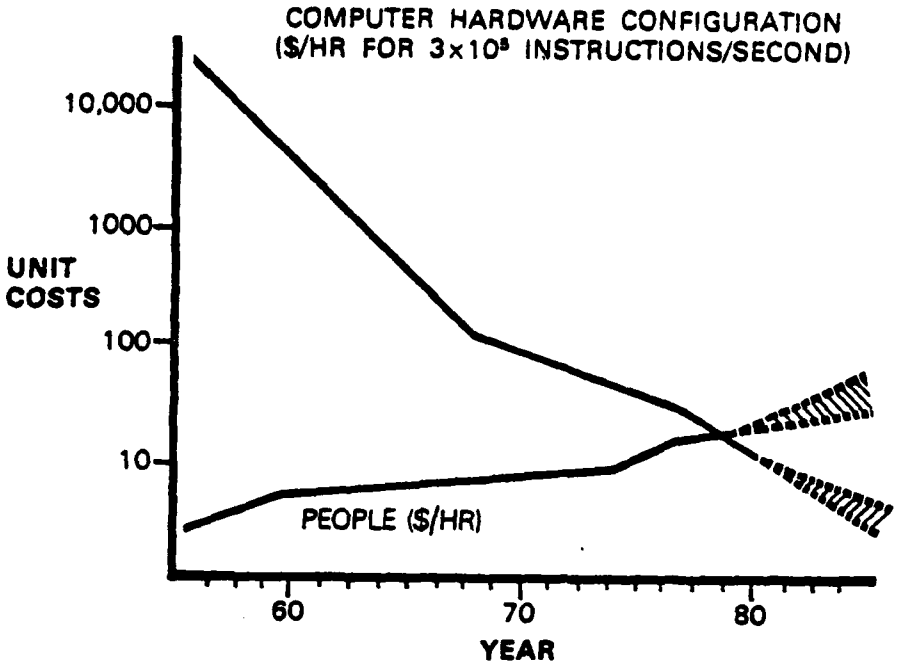


Figure 1-1. Hardware/Personnel Processing Costs

The life cycle of computer programming software is categorized into two fundamental phases: development and maintenance. Figure 1-2 illustrates where the software dollar is spent during the life cycle. The development phase consists of the analysis, design, coding, testing and implementation of computer software which meets specific user functional requirements. The maintenance phase consists of removing software defects and adding software enhancements. About 70 percent of the overall cost of software is spent on software maintenance functions (Ref

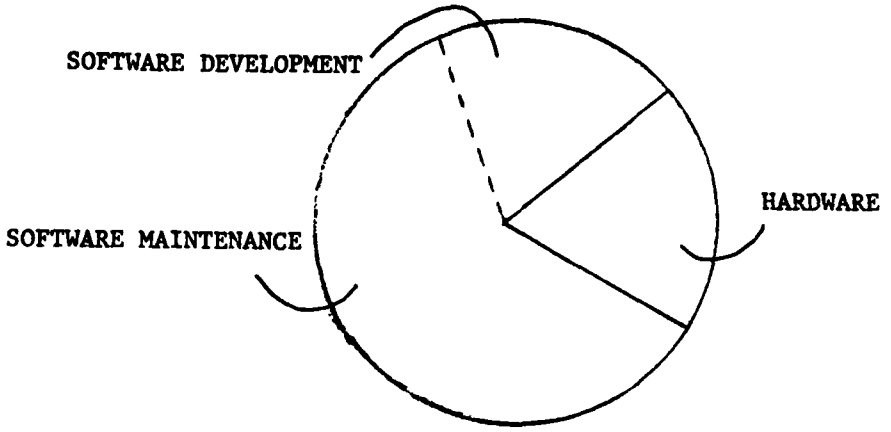


Figure 1-2. Software Dollar

10:65). Figure 1-3 graphically reveals that as software costs grow, a larger proportion of software dollars are going to software maintenance than to software development. This trend will continue as long as increasing amounts of software are added to our inventory and the operational life span of the software tends to get longer. The result is more software in the maintenance phase of the life cycle. In addition to the monetary commitments, 75 percent of the programmer workforce is committed to software maintenance activities instead of developing new applications (Ref 68:87). The current situation has caused severe budget restrictions and controls to be instituted. For this reason. the current large investment of resources and the projected future costs have caused widespread concern.

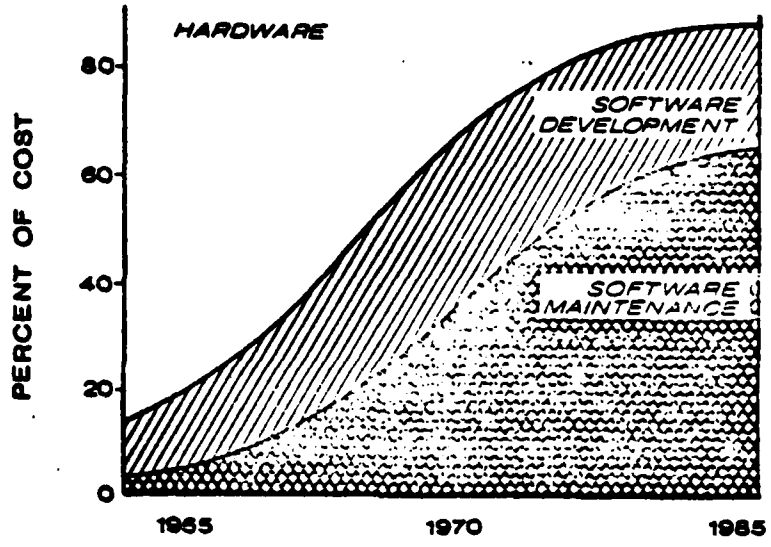


Figure 1-3. Hardware/Software Cost Trends (Ref 9)

Research Scope - Weapon System Computers

The impact of the explosive growth of software is also being felt by Air Force weapon system computer applications. Indeed, the life cycle cost of computer software is becoming a major component of Air Force weapon systems. In 1977, the Department of Defense (DOD) spent over $3 billion on defense system software (Ref 48 and 74) and about $4 billion in 1978 (Ref 78). In terms of costs of acquiring weapon systems computer systems, 80 to 90 percent goes for software. For example, in 1976 the total cost of software in the Worldwide Military Command and Control System (WWMCCS) was estimated to be $722 million dollars or ten times the cost of the WWMCCS hardware (Ref 74). Software represents 4 to 5 percent of the Air Force

budget and almost 6 percent of the NASA budget. These expenses are predicted to increase dramatically. A recent Electronics Industry Association DOD Digital Data Processing Study was performed by an industry team consisting of representatives from IBM, INTEL, Control Data Corporation, ROLM Corporation and TRW. The result of their study was a ten year forecast of Embedded Computer Systems Workload in terms of number of computers (Figure 1-4) and hardware and software costs (Figure 1-5). Note that these growth predictions parallel the estimates made in the first paragraph of this thesis.

As increased investments in the development of Weapon System Software have been made, the cost of supporting and maintaining these systems has become substantial. Barry DeRose and Thomas Nyman (Ref 14) verify this situation with predictions of what it may take to support the life cycle of future Weapon System computer systems.

> "The majority of complex software systems are new and still in the development cycle. As these new systems are deployed, this cost distribution will reverse to emphasize the increased O&M burden. This, coupled with greater system longevity, may ultimately result in a five or ten to one ratio of O&M cost to R&D cost when viewed over the total life cycle of a typical system. With these projections, we will need an army of software maintainers."

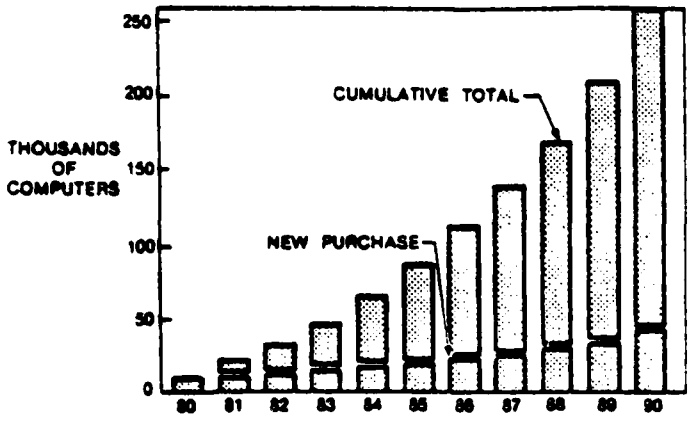


Figure 1-4. Ten Year Forecast: DOD ECS

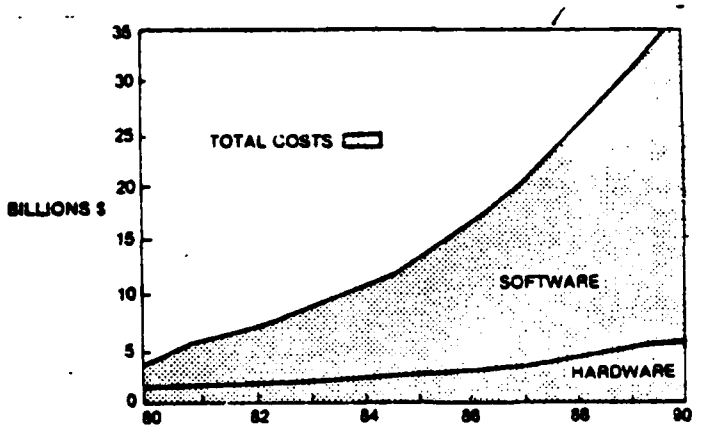


Figure 1.5. Ten Year Forecast: Hardware/Software Costs

Maintenance of weapon system software takes up a significant portion of the life cycle costs. For example, a study conducted by Barry Boehm (Ref 9) found the maintenance portions of the 10 year life cycle costs for two Air Force command and control software systems to be 67 percent and 72 percent of the total. Note that this agrees with the findings for all systems as displayed in Figure 1-

2. Indeed, the cost of completing maintenance tasks is many times that of development tasks. A DOD study has shown that the development costs for Air Force avionics software averaged about $75 per instruction while the costs of maintenance changes were in the range of $4000 per instruction (Ref 98).

The high cost of the maintenance activity is drawing increasing attention. Both management and research efforts have been initiated to investigate and reduce the high price tage associated with software life cycle maintenance.

The rising cost of software maintenance is affecting Air Force computer systems. Within the Air Force, computers are administratively categorized into two functional partitions: Management Information Systems and Weapon System Computers. Typically, Management Information Systems are general purpose, commercially available computers utilized for accounting and finance, inventory supply management, personnel control, training, research and development or intelligence data gathering. Weapon System computer systems tend to be special purpose computers (sometimes one of a kind) that play an integral part in the operation of a deployable weapon system. Weapon Systems computers support functions such as command, control and communications, weapons delivery, avionics navigation, and radar. The main thrust of the research and investigation of this paper is directed at Air Force weapon system computer systems and not Air Force management

information systems.

Current Perspective Weapons System Software Management

The early 1970's marked the beginning of the deployment of weapon systems where weapon system computers played an integral part in the total system structure. During this time period, problems occurred with the development and support of the computer software. Several defense sponsored studies (Ref 32,42,57 & 66) revealed these problems and suggest areas where defense system software required improvement. Excessive development and maintenance costs, lack of standardization, scheduled slippages and delays, and excessive software errors were the problems which were identified. The studies pointed out specific management practices which required major rework in order to reduce system costs and increase software reliability. Sample recommendations included:

1. Modification of Procurement and Configuration Management Policies,
2. Utilization of new software engineering practices in order to increase productivity and maintainability.
3. Use of standard well established program languages,
4. Control of software support to the same degree as hardware components.

In 1976, the Department of Defense responded with a Software Management Plan (Ref 15) which outlined steps to decrease the reported software problems. DOD Instruction 5000.31 was also released in order to establish seven high order languages as interim standards for use in developing new defense systems. Also DOD Directive 5000.29 was

released in order to establish policy for software management of embedded computer systems during development, acquisition, deployment, and support. In addition, a DOD Software Science and Technology (S&T) program was initiated in order to investigate additional methods for curbing the software costs. It was understood that further research and development was necessary to increase management and programming productivity. Several activities were initiated with this objective in mind. The results of some of these activities will be discussed in Chapter II.

Following the lead of the DOD initiatives, the Air Force has instituted several policies and procedures for the management and control of the life cycle of computer software. Table 2 summarizes the documents which influence the development and maintenance of weapon system software. Additionally, several guidebooks (Ref 1, 18-20 & 93) have been developed to aid program managers and programmers in such areas as life cycle planning, documentation standards, structured programming, configuration management, software quality assurance and software maintenance. This guidance has offered assistance to software managers by outlining management methods which have proven successful in past weapon systems.

The major theme which prevails through the change in DOD and Air Force policy initiatives is to manage software more like hardware throughout the life cycle. Implementing these programs has provided a defined, formulized approach

to controlling the development and maintenance of weapon system software. Yet, software management difficulties still persist. An extensive amount of study has been directed at perfecting management reviews and programming technologies within the software development phase. There has been limited study on how to improve the factors which influence maintenance productivity. This thesis will extend the boundaries of previous studies by investigating the software maintenance process within Air Force Weapon System programs.

Table 1 - Regulation Documents for Air Force Weapon System Computers

Document	Function
DODD 5000.1	Major Systems Acquisition
DODD 5010.19	Configuration Management
DODD 5000.29	Management of Computer Resources in Major Defense Systems
AFR 800-14 Vol I	Management of Computer Resources in Systems
AFR 800-14 Vol II	Acquisition and Support Procedures for Computer Resources in Systems

Problem Description - Management of Maintenance Costs:

Software maintenance managers are feeling the impact of having to support the increasing software inventory. This situation coupled with the fact that the life span of software is increasing has caused the software maintenance phase to become a significant activity for managers and programmers.

Current literature has indicated that there are many factors which are causing difficulties within this newly expanding phase of the software life cycle. One report suggests that the major problem lies in the way management views the maintenance process. On February 1981, a GAO Report to the Congress titled "Federal Agencies' Maintenance of Computer Programs: Expensive and Undermanaged" summarized a maintenance - management problem which persists within several Federal Agencies. The Comptroller General stated:

> "In spite of the high cost, agencies have a very limited overview of their software maintenance operation and have made little concentrated effort to effectively manage and minimize the resources required to maintain their computer software.
>
> Maintenance is not managed as a function. That is, ADP managers have done little either to identify common causes of maintenance problems or to take action to reduce maintenance costs. The absence of maintenance management is due in part to (a) the absence of a uniform definition of maintenance, and (b) the absence of Governmentwide guidance on how to control software maintenance and reduce its costs.
>
> Managers generally have neither cost accounting data nor management data on software maintenance activities and thus know little about how much maintenance really costs overall, or which types of maintenance cost the most."

Current Air Force regulations and guidelines do not adequately address the software management procedures necessary to limit expenditures while maximizing the usage

of available productivity methods. The first step towards controlling maintenance costs is the identification of the factors which are causing problems. This would allow management to take action to reduce maintenance resource expenditures by reducing the problems which are causing excess costs. Barry DeRose (Ref 14) further indicates that these first steps have not been accomplished, when he states:

> "DOD currently suffers from a poor historical data base of software costs. Not only is the DOD unaware of what is being spent on software in the development, production, and operational/maintenance phases of a defense systems' life, but there is considerable uncertainty as to the proportions of dollars which should be spent. As it currently stands, DOD in general does not budget dollars or plan time for the predictable problems and changes which we should prudently anticipate in each of these phases."

The high cost of software is a system of a situation out of control. Since maintenance functions are increasingly consuming more of the software dollar, control efforts should be aimed at this phase of the software life cycle. Where should these control measures be directed? This thesis will take a step towards answering this question.

Research Objectives

The purpose of this thesis was to study the software maintenance process of Air Force weapon systems in order to identify ways to reduce software life cycle costs. This was accomplished by meeting the following objectives:

1. To identify the factors which influence the level of maintenance costs.
2. To identify Air Force policies which tend to cause excess resource commitments to the software maintenance of weapon systems.
3. To develop policy recommendations to improve maintenance support of existing and future software programs.

By identifying the factors which influence maintenance resource expenditures, managers will be able to evaluate decisions which affect the level of these commitments. In addition, if problem areas can be identified and resolved through policy changes, managers and programmers can spend more time and effort on maintenance tasks.

Research Methodology

Completion of these objectives was accomplished in three phases. The first phase involved a general literature review in order to establish the software maintenance life cycle within Air Force weapon systems. This was accomplished by defining general maintenance concepts and then applying them to weapon systems through the summary of recent software maintenance studies. Chapter II provides this description.

Because of the lack of current and detailed information, the second phase involved first hand investigation of the software maintenance process within Air Force Weapon Systems. This was accomplished by conducting a structured interview of the managers and programmers from twelve Air Force Weapon System software

projects. Chapter III describes the methodology used for this investigation.

The third phase involved the analysis and application of the findings of these interviews. Chapter IV provides a comprehensive analysis of the data gathered. Chapter V provides recommendations for improvement of the software maintenance process within Air Force Weapon Systems.

II. Software Maintenance - A State of the Art Review

The increasing cost of software maintenance has been the major impetus and justification for recent software research efforts. Software maintenance research in the last decade has progressed in three directions:

- Definition of the maintenance process
- Identification of problem areas within the maintenance process
- Development of management and programming methods for solving maintenance problems.

The vast majority of the progress that has been made in these areas has been supported by government sponsored studies. This chapter will summarize current accomplishments and establish a basis for investigating the maintenance life cycle of Air Force weapon system software.

Software Maintenance - An Overview

While reviewing the literature, several descriptions of software maintenance were encountered (Ref 6,11,29,31 & 93). In hardware terminology, maintenance is defined as "repair or restoration to the state prior to failure." (Ref 68) Softwa does not deteriorate or degrade over time. Any faults that are detected during the maintenance phase are errors that were made previously in the life cycle. Swanson's description of the maintenance of software has gained popular acceptance. (Ref 95) He identifies three

classes of changes that occur during maintenance:

1. Corrective - Fixing a pre-existing error.
2. Adaptive - Modifying the software to accommodate environmental changes and requirements.
3. Perfective - Improving or augmenting the performance capabilities.

The process of making these changes is diagrammed in Figure 2-1. Maintenance tasks are identified by analyzing the reported software deficiences and requests for software enhancements. The analysis consists of the validation of submitted requests and implementation feasibility studies. The result of this process is the prioritization of software change requests. Once a maintenance change is identified, the process of developing the indicated function begins. This development utilizes the traditional approach for software production: analysis, design, code, testing. The analysis phase consists of understanding the change functions and the software to be modified. The design phase consists of a redesign of the software taking into consideration the possible ripple effects of making changes (Ref 106 & 107). The coding phase implements the program design using a set of software maintenance development tools. The test phase assures that the developed code runs as specified. Then, the next major step is to integrate the code modification into the baseline system and run system tests. The system check-out usually consists of laboratory

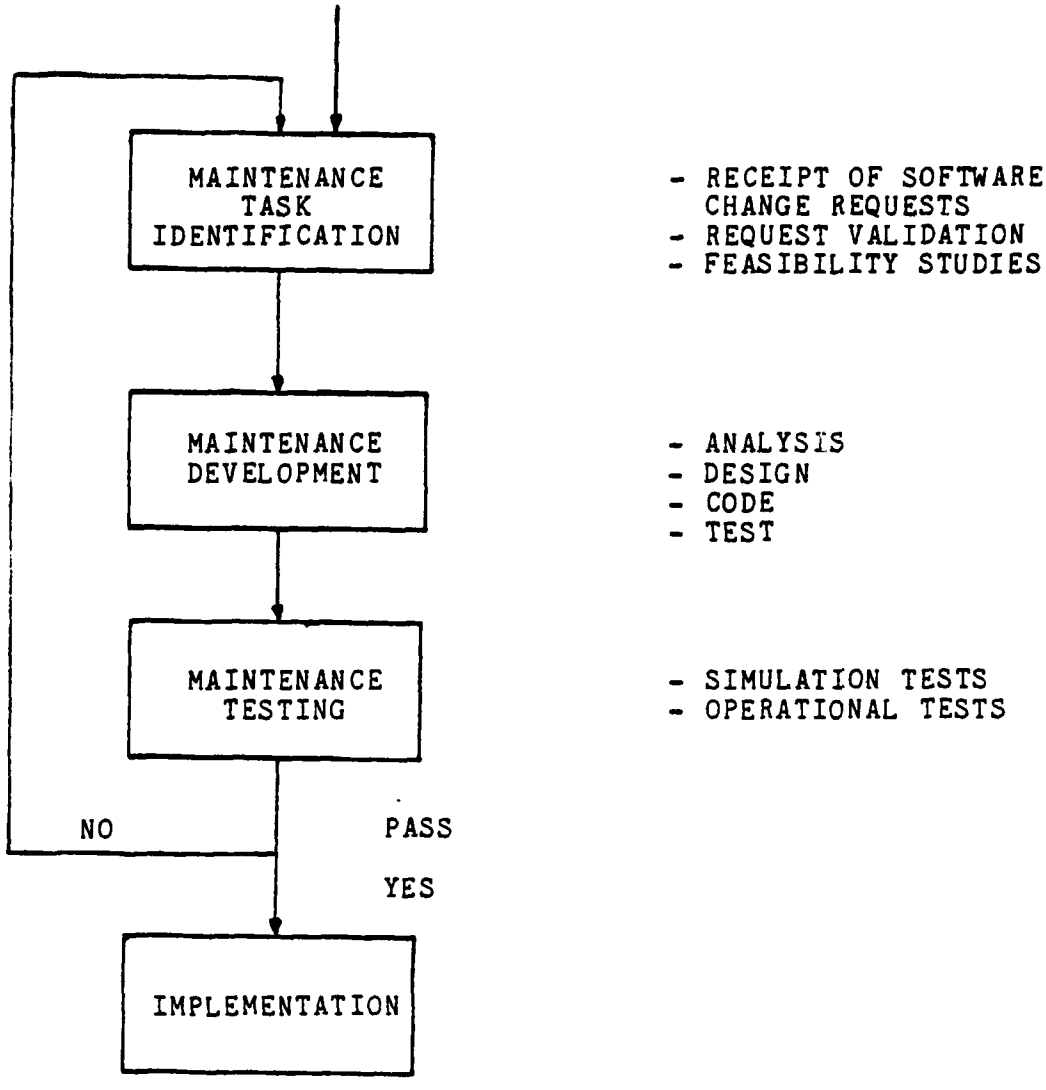


Figure 2-1. Maintenance Process Functions

simulation tests and operational tests. If any errors are identified during operational testing then the maintenance process starts again; otherwise, the code change is implemented and documentation is altered.

Management Approach

The maintenance process as illustrated by Figure 2-1 is affected by management controls in several ways. Within most organizations the software maintenance management controls are packaged together under one title - configuration management. The objective of this section is to split-out the various management functions and describe how they affect the maintenance process.

Readiness Planning

Management of a system which is planned to change, requires the planning and development of resources so that the organization is prepared to respond to the change. Several methods are used to estimate the time, personnel, and other resources required to support software maintenance functions. Good judgement and experience coupled with several quantitative methods have aided project managers with the prediction task. Several life cycle cost models are available: RCA PRICE - S (Ref 90), TRW SCEP (Ref 1), IBM Walston - Felix Cost Model (Ref 100, and Putnam SLIM Ref 86 & 87). It has been discovered from comparing the life cycle costs of software projects, that there is a basic pattern of resource utilization. The input to these costing models is the project manager's estimations of the factors which best describe the software product before and after development (size, application type, development environment, etc...). The output is a projection of

software modification and maintenance costs during the operational life of the system.

Another method of maintenance workload prediction involves the use of software reliability studies. As was first accomplished for hardware reliability, techniques have been developed that estimate operational reliability of the software over the system life. By using attributes of the developed system, the mean time to next failure (MTTF) and mean time between failures (MTBF) can be estimated. This gives managers indications of how often corrective maintenance functions may have to respond to software error detection. Several reliability models are being developed and are available for limited application: Littlewood Bayesian model (Ref 54 & 55), Schick and Wolverton model (Ref 91), Musa (Ref 75), Trivedi (Ref 99) and Okumoto - Goel model (Ref 77 & 78).

Once the rate of the program correction and user enhancement requests can be estimated, the next step is to judge how easily the software can be modified. Several software quality metrics have been developed which can be used to evaluate the level of software flexibility, portability, and reuseability. Boehm et al (Ref 11) and McCall/Walters (Ref 62) have identified quantitative metrics for quality evaluation. Halstead (Ref 35) and McCabe (Ref 59 & 60) have developed methods for analyzing program complexity. Management has applied quality metrics to the evaluation of software maintainability. The Air Force Test

and Evaluation Center (AFTEC) performs operational test and evaluations of system software as part of a complete Air Force weapon system assessment. During the evaluation, software maintainability is a primary objective. Computer program code, supporting documentation, and the software support system are evaluated based on characteristics which affect the capability of support personnel to accomplish software maintenance. Numeric scores are assigned to each of the maintenance characteristics. In order to evaluate these scores, AFTEC has assigned threshold, and goal scoring values. Evaluation scores falling below the assigned threshold value indicate deficiencies while those falling between the threshold and goal are judged satisfactory and those exceeding the goal are considered excellent. A beneficial result of this methodology is that it produces a very specific list of items that need improvement. The evaluation methodology is based on weighting those factors which are thought to most influence the ability to support the operational software.

Estimating resources to support software change requirements is an ongoing process which ultimately affects other management controls of the maintenance process.

Change Control

The maintenance process is initiated when a request is made to change the software product in order to correct detected errors, perfect performance or adapt the software

to other applications or environments. A major management problem is the control of these changes. The objective is to:

1. make only needed changes,
2. insure that changes are effectively integrated and tested, and
3. insure that documentation is changed to remain compatible with the software.

Several DOD and Air Force documents (Ref 1,16-27) have been developed to outline procedures for completing software configuration management. Figure 2-2 illustrates the role configuration management has in identifying tasks within the maintenance process. As summarized earlier, a series of committee reviews evaluate proposed software changes. One of the considerations of change approval is the level of resources that are currently available. This is often a direct result of previous readiness management predictions and planning. Just as readiness management functions influence configuration management, change control decisions often influence future resource requirements.

The final result is not only the identification of which software changes can be completed but which ones are to be accomplished.

Quality Assurance

Software quality assurance objectives differ from organization to organization but usually entail the review,

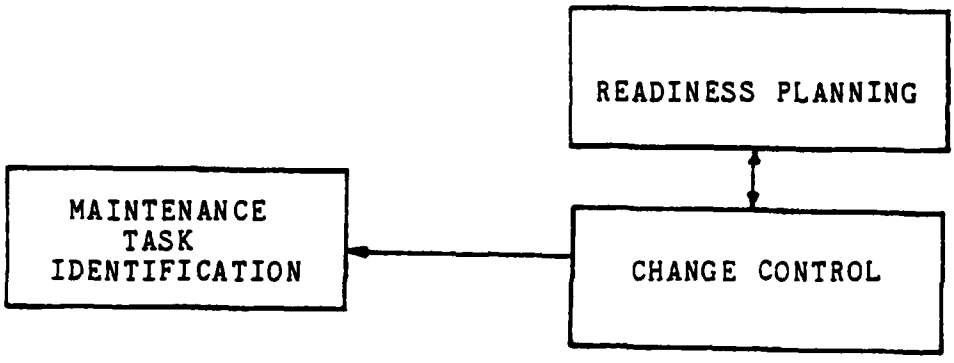


Figure 2-2. Configuration Management Influence

rewriting, and enforcement of standards or conventions which affect the quality of software code and documentation. The philosophy of quality control has been borrowed from material manufacturing and production applications and has been aplied to the development of software. Figure 2-3 illustrates how the maintenance and development phases are influenced by management's attempt to enforce design, coding, documentation, and testing standards upon programmers. Several DOD programming standards have been developed to support quality assurance efforts. Guidebooks have been established to suggest methods of applying these standards (Ref 1 & 93). In effect, what is being attempted is the integration of perfective maintenance concepts into all types of software changes. The result has been increased software performance, reliability, and maintainability. For example, after implementation the suggestions of a quality assurance program, the cost of executing one application was reduced 60 percent and the

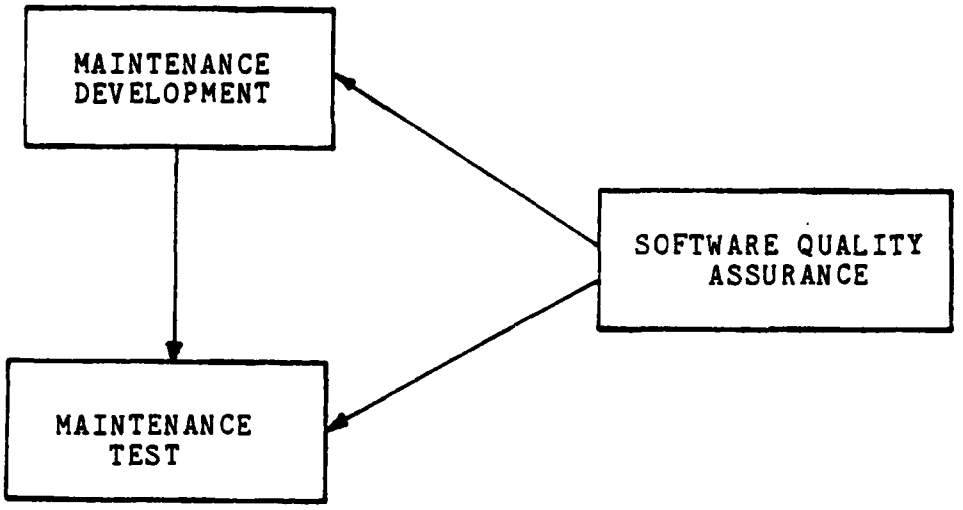


Figure 2-3. Quality Assurance Effect on Maintenance Functions

execution time was slashed 77 percent (Ref 34). In addition, the occurrence of error within the system was significantly reduced. These results were obtained primarily from the expense of the labor that was required to produce qualitative documentation and program code. The success of quality assurance programs rests on the level of funding and staffing support. as well as, the tenacity with which management pursues the program's objectives.

Scheduled Implementation

Scheduled implementation of changes to the software configuration provides a controlled structured integration environment. Software modifications are ground together and implemented in scheduled baseline phase releases. Lindhorst (Ref 53) indicates several benefits to this management

approach:

- Consolidation of requests. Some efficiency can be achieved because multiple changes to the same program or module can be combined under one maintenance task.

- Programmer job enrichment. The maintenance schedule should provide an opportunity for selective programmer upgrade training or career broadening assignments.

- Forces user department to think more about the changes they are requesting. Delayed implementation of new capabilities will tend to filter out those changes that will be short lived, unimportant or both.

- Periodic application evaluation. Scheduled changes provide convenient milestones for consideration of the cost effectiveness of continuing the current system.

- Elimination of the "squeaky wheel syndrome". When users realize that change requests all receive equal consideration and implementation of the changes is on a planned basis, there is less cause for attempting to pressure the maintenance staff.

- Programmer back-up. The maintenance staff manager has more latitude in assigning his personnel to tasks and can conduct crosstraining within the maintenance teams.

- Better planning. Long and short range staff planning can be more effectively accomplished when the workload can be predicted with a reasonable degree of accuracy.

- Data processing change requests are regarded as being as important as user requests. Under this type system it is possible to give both user and change requests fair consideration when planning for the next scheduled maintenance period.

Figure 2-4 summarizes the effects of scheduled maintenance and other management approaches to controlling and influencing the maintenance process. Scheduled maintenance dictates when approved software changes will be

released to system operations. Quality assurance programs assure that all software changes within a phased release conform to a level of performance and programming standards.

In summary, management attempts to assure that there are sufficient maintenance resources to complete validated software changes. During the implementation of the changes, management is attempting to reduce the propagation of future constraints which may cause recurring maintenance. The next section will review the programming methods which influence the accomplishment of assigned maintenance tasks.

Programming Approach

The literature abounds with methods of designing, building, and documenting software systems. The most noted are structured design (Ref 108), the Jackson method (Ref 73), the structured analysis and design technique (SADT) (Ref 13), the Warnier-Orr Approach (Ref 101), the system design methodology (SDM) (Ref 36), architecture definition technique (ADT) (Ref 4), and the requirements engineering validation system (REVS) (Ref 39). These methods are intended to increase the overall productivity of the programming staff and can be applied to software production during the development phase or maintenance phase. Test procedures such as structured walkthroughs have also been developed to assist programmers during software production. Figure 2-5 outlines how the software development and test methods influence the maintenance process.

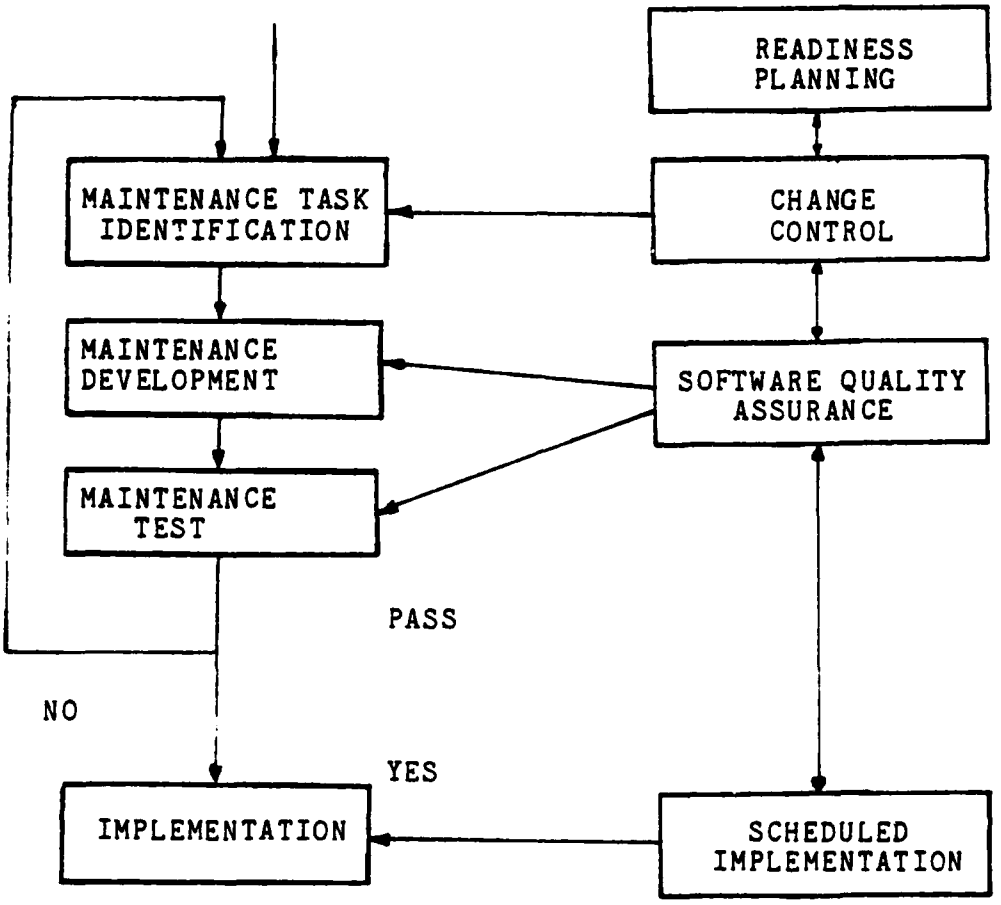


Figure 2-4. Management Influence on Maintenance Process

Automated tools have been developed to assist with management and programming functions. For example, computerized costing models are used to assist managers with resource requirement predictions; debugging and diagnostic packages assist programmers with removing detected errors from software. John Donahoo and Dorothy Swearinger (Ref 29) have categorized all the existing software tools as they relate to specific managment and

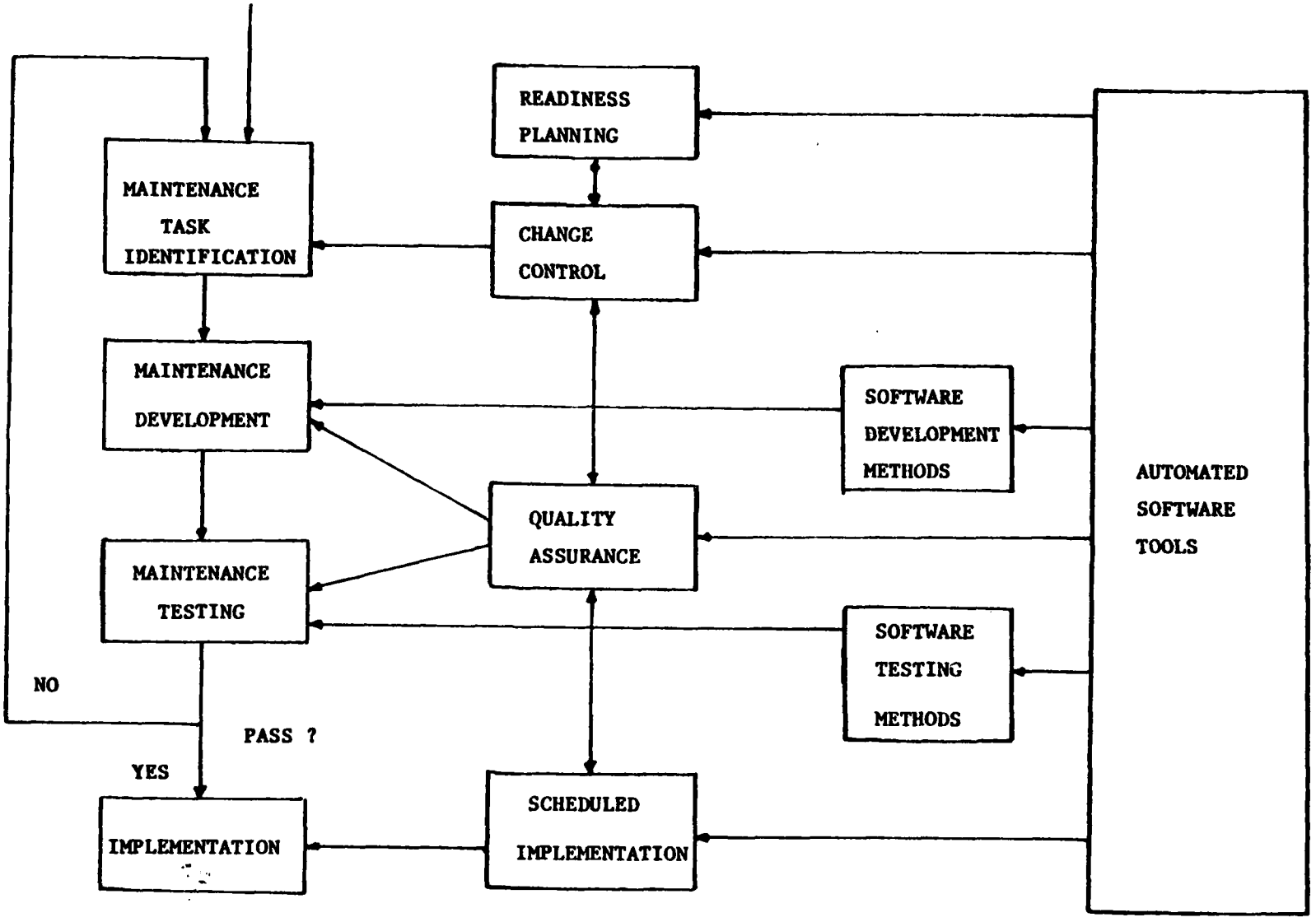


Figure 2-5. Management and Programming Influences on the Maintenance Process

programming functions within the maintenance process. In addition, the BDM study has investigated the extent to which these tools are currently being utilized within military weapon system applications. Air Force guidebooks (Ref 1 & 92) have been developed to discuss how these automated tools can be utilized to perform maintenance activities.

Previous Studies

Four studies have provided the majority of the direct investigation of the support process of software systems. Each study was completed with separate research objectives and therefore analyzed the support process from a different perspective. As a result, we begin to picture the structure of the software maintenance process of Air Force weapon system computer applications. The findings of each of these studies are provided next.

Study 1: Lientz - Swanson Study - 1978 (Ref 46-52)

The first extensive studies of maintenance functions were a series of surveys conducted by the UCLA School of Management. The most comprehensive of these surveys is the study completed by Lientz and Swanson in 1978.

The problems of maintaining application software within 487 data processing organizations were surveyed. The survey investigated the gamut of business applications. The typical application software system had the following attributes:

1. Average Operations and Maintenance age of 3 years 4 months.

2. Consisted of 23,000 lines of code in 55 program modules.

3. 75 percent were written in COBOL or RPG.

The maintenance process for these applications was studied with the following results:

1. Through the maintenance phase, program code grew in size by 10 percent per year.

2. About 75 percent of the systems were maintained by the equivalent of one programmer or less.

3. Maintenance time was allocated to tasks as follows: 20 percent - corrective; 25 percent - adaptive; 55 percent - perfective.

The magnitude of the maintenance effort was found to be affected by four variables: system age, system size, relative amount of routine debugging, and the relative development experience of the maintainers. Figure 2-6 illustrates the paths that interrelate these variables. Five causal paths relate the variables in the diagram. The first path shows that as system age increases so does system size. The increase in both variables leads to greater level of maintenance effort. The second path is an extension of the first. This path indicates that increases in system size cause increases in the relative amount of routine debugging and maintenance effort. The third causal path shows that with increased system age, the relative experience of the maintainers tends to decline leading to increased maintenance effort. The fourth path indicates that as the relative experience of maintainers declines, the amount of routine debugging and maintenance effort

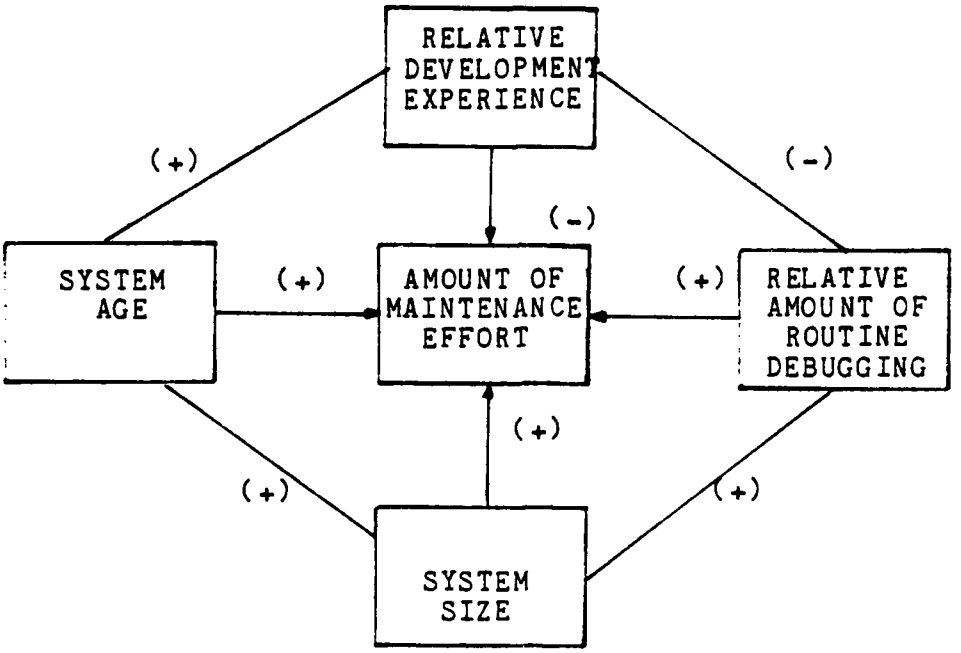


Figure 2-6. Causal Effects on Level of Maintenance

increases. The fifth causal path indicates that increased system age leads directly to an increase in the level of maintenance effort.

Where development tools were used, the software product was perceived to be of better quality. But there was little evidence to show that either development tools or organizational controls contributed to a reduction in maintenance person-hours. Factor analysis of survey results identified six major problem factors with the maintenance process:

1. User knowledge

2. Programmer Effectiveness
3. Product Quality
4. Programmer Time Availability
5. Machine Requirements
6. System Reliability

This study was the first attempt at specifically studying the software maintenance process with business applications. The major results of this study were the identification of the problems within the process and a correlation of several factors with maintenance level of effort. This study did not look at military real time processing applications and did not identify policy decisions which caused management and programmer problems.

Study 2: Lientz - Wegner Study - 1980 (Ref 52)

At the request of the Department of Navy, Lientz and Wegner applied some of the earlier maintenance study objectives. This study was directed at 18 Navy and Marine Corps weapon system software projects at eight different locations. The major purpose of this study was to assess the current state of the development and support process of Navy weapon system software in order to provide a basis for improving cost, reliability, and performance. The typical application software project that was surveyed had the following attributes:

1. Average operations and maintenance age of six years.
2. consisted of 176,000 lines of code within 115 program modules.

3. Over 80 percent were written in Assembler or CMS-2.

The Lientz-Wegner study concluded that:

1. Program code had a yearly growth rate of 5 percent
2. Program code is usually developed by one contractor and maintained by another.
3. These applications performed more test and verification than business applications because of the requirement of high operational reliability.
4. Perfective maintenance is allocated the majority of available maintenance man hours.
5. The majority of weapon system applications utilized formal quality assurance and configuration control programs.

The sample size of this survey disallowed the type of statistical analysis performed in Study 1. Yet, this study did identify three conditions which are perceived as major causes for software change of modification: hardware change, user demands and development of new weapon systems.

The major result of this study was a profile of the maintenance life cycle of military software systems and a checklist of management and programming methods and tools being used. The study did not look at Air Force weapon system software applications and did not directly investigate policy decisions which affect excess maintenance costs.

Study 3: BDM Study - November 1981 (Ref 97)

The objective of this study was to identify and describe methods for evaluating the software support facilities being used to maintain weapon system embedded computer software. BDM corporation developed recommendations based on the observations of the maintenance

process and the automated tools being used at 12 military locations. BDM found that the maintenance process consists of similar functions within each of the systems they observed. The process of modifying weapons system computer software involves these steps: analysis of change requirements, redesigning, programming, debugging, integrating, testing and verifying. Formal controls have been established for managing software changes. Figure 2-7 illustrates how software is managed in conjunction with changes in weapon system hardware. The process begins with the reporting of system problems or change requirements to the program manager. This results in a study of the validity, feasibility, risk, costs, and impacts of implementing the requested changes. The results of this study are then evaluated by the Configuration Control Board (CCB). This board analyzes the cost/necessity issues and allocates resources to implement needed changes. The Computer Program Configuration Sub Board (CPCSB) reviews the development of software to insure correspondence with schedules, requirements and resource limitations established by the CCB. Detailed configuration management procedures define the interface responsibilities between these change control boards.

During "maintenance development", the change requirements passed by the CCB are translated into a detailed redesign of a portion of the software. This design is coded and eventually assembled into object code. The

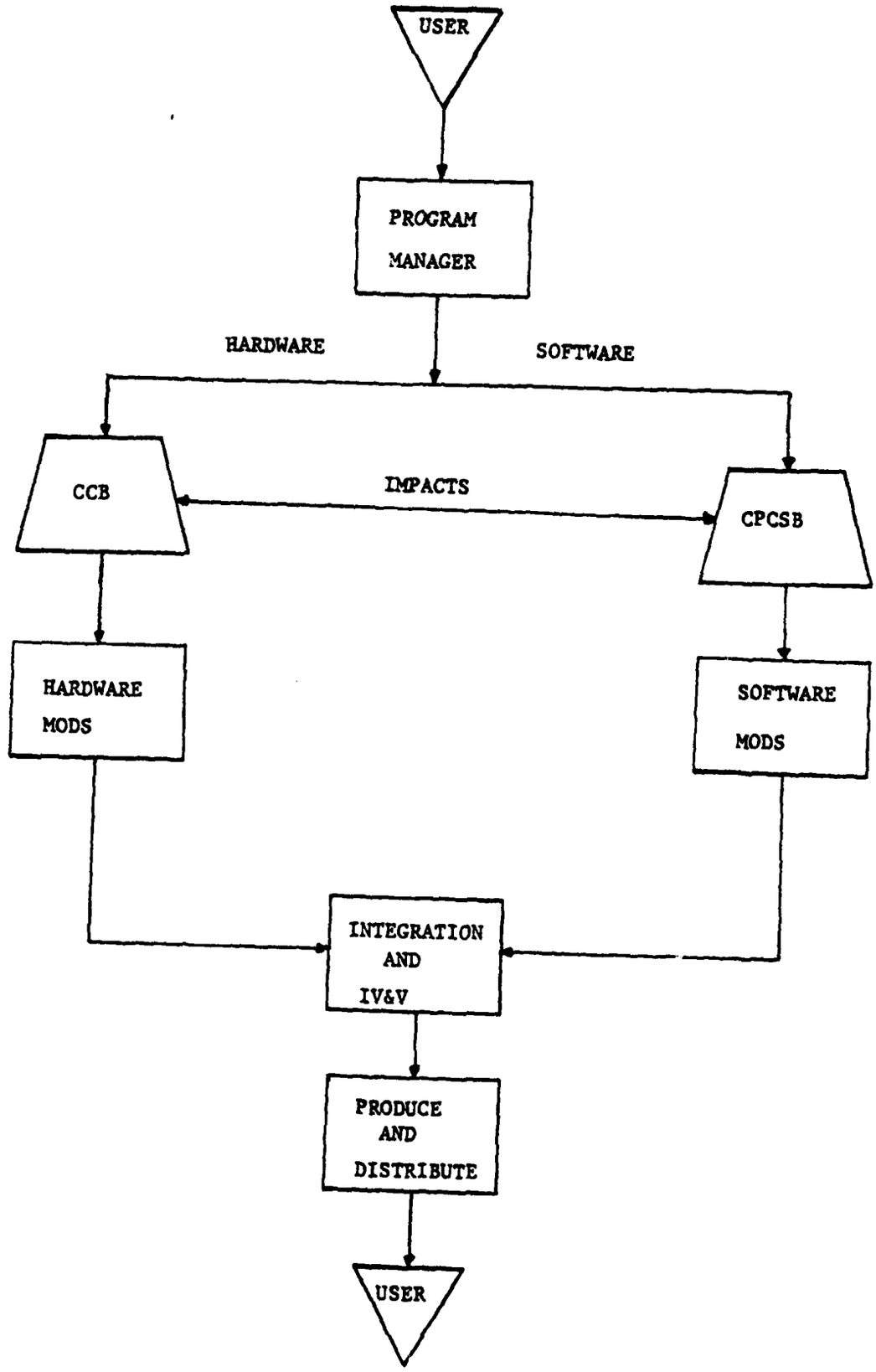


Figure 2-7. Configuration Management Organization

object code is then tested and validated before it is implemented into the target computer system. The programming, integration, testing, and validation are performed on automated support systems. BDM partitioned and described these support systems in six categories:

1. Host Processing Systems
2. Software Benches
3. Laboratory - Integrated Test Facilities
4. Operational - Integrated Test Facilities
5. Configuration Management Systems
6. Other Support Systems

Host processing systems can be the target computer or a separate larger development system. The larger development computers are typically equiped with an operating system, compilers, assembler, editor, program maintenance library software, and debugging packages. These features facilitate large scale, multi-user development. Software benches use a simulation CPU to represent the target processor external environment and interface with either the actual target processor or an instruction level emulation of the target processor. Laboratory-integrated test facilities exercise the operational software on the target computer in a simulated operational environment. Most of the equipment which is tied to the target computer are used in these tests. Operational-integrated test facilities exercise the software within an actual or representative operational environment. In order to assist with the control of

software changes there are a limited number of automated configuration management tools. Yet, most configuration management is still being accomplished manually. Various other support systems are being utilized such as automatic documentation generators, project planning and tracking systems, and word processing systems.

Study 3 provided a more descriptive look at the inner workings of management controls and programming development tools within Air Force, Navy, and Army weapon system software. The purpose of the BDM study was not to take an analytic approach to improving software maintenance conditions within these systems. An attempt will be made by this thesis to develop improvement recommendations.

Study 4: General Accounting Officer (GAO) Study - 1981

The GAO reviewed computer software maintenance within 15 federal data processing installations. The purpose of the study was to determine whether the problems found in certain sites existed at other federal organizations. Their findings strongly indicated that several problems are wide spread. The GAO reports that of the systems investigated two-thirds of the programmers were assigned to maintenance functions and approximately 80 percent of maintenance costs are expended on paying salaries for maintenance personnel (managers, analysts, and programmers). As might be expected, the remainder of their findings were directed at emphasizing deficiencies in the way softare maintenance is

being performed. The most significant findings were:

1. Government managers are not managing software maintenance as a function.
2. Government managers do not have available specific standards and goals for performing software maintenance management functions.
3. The documentation of systems is either inadequate or missing.
4. Software support tools are utilized only on a limited basis.

Because of the vast range of applications which were reviewed, a detailed description of the surveyed software systems was not reported.

The General Accounting Office Study reaffirms the general findings of the previous studies and provides a mandate for further investigation. The findings of these studies have provided much of the information needed for understanding and managing the resources involved in the support of Air Force weapon system software. The current literature was further investigated in order to formulate further hypotheses by which Air Force weapon system software could be evaluated. Several articles reported maintenance problem conditions which are causing managers and programmers to be unproductive. These maintenance problem conditions are discussed below.

Maintenance Problems

The following maintenance problem areas have been identified:

- High turnover rate and low level of personnel experience
- Low level of software reliability and quality
- Lack of maintenance standards
- A misdirected management control system
- Poor software documentation

Most sources agree that maintenance tends to be a personnel intensive activity. Typically, 75 percent of the programmers in the programmer pool are directly involved with the maintenance function. This situation coupled with the rising cost of programmer salaries has caused personnel costs to be the most significant cost factor of the software maintenance dollar. The probable causes for the personnel intensive condition have been discussed by several authors. Liu (Ref 56) has found that when a person is assigned to a maintenance group, tasks are assigned without adequate training or proper orientation. To make things worse, there are rarely proper maintenance standards or procedures upon which to refer. As a result, the individual is left to develop a unique set of "tricky games" in order to accomplish maintenance tasks. Thus, it often takes two to three years to train someone to assume responsibility for a large application system. Then, just as programmers gain their experience with a system, they are lost by job

turnover or career progression. Canning (Ref) reports that personnel turnover is a serious maintenance problem. Indeed, Boehm (Ref 9) has found that maintenance programming has been given more than secondary attention. In fact, many managers still look upon maintenance programming as a necessary evil and tend to assign the junior programmers to fill the positions. Because of the emphasis on new systems, programmers working on development tend to receive greater career visibility and progression. Under these conditions no wonder programmers seek refuge from that so called "drudgery" of maintenance work.

Maintenance programmers tend to work under great management pressure to fix software according to set schedules. As a result, getting the task done becomes the prevailing theme, while maintaining software quality sometimes becomes a luxury consideration. Through the study of many system life cycles, Lehman (Ref 44) has found that as software systems get older they tend to get increasingly complex unless something is done to simplify or restructure. With the lack of experienced, capable programmers, emergency corrective type fixes and "greatly needed" user adaptive extension take first priority over perfective improvement tasks. As a result, quality factors such as system documentation are often times infinitely delayed (Ref 56). Several writers have indicated that documentation is a critical problem during maintenance (Ref 6,44,49 & 85).

As corrective, adaptive and perfective changes are made to software, additional errors are induced into the system. McHenry (Ref 52) reports that the probability of begetting errors during this period is normally 10 percent. Yau and Collofellow (Ref) indicate that a ripple effect condition often causes changes in one module to cause errors to ripple through to other modules. These undetected errors cause further maintenance effort. Figure 2-8 (Ref 63) shows how these undetected errors increase corrective maintenance effort later in the life cycle. As corrective maintenance work is accomplished, the quantity of undiscovered errors increases. Then as the software is operationally exercised through logic scenarios not earlier tested, these errors are discovered creating more corrective maintenance. Without controlling the rate of error during software changes, this cycle will cause corrective maintenance to grow exponentially through the life cycle of the software.

The objective of management is to control the instability of this process and at the same time increase the productivity of personnel and computer resources. Yet, historically maintenance management efforts have been by the "seat of the pants." Several writers (Ref 30) have identified the management control system as the weakest link and most deserving of increased attention.

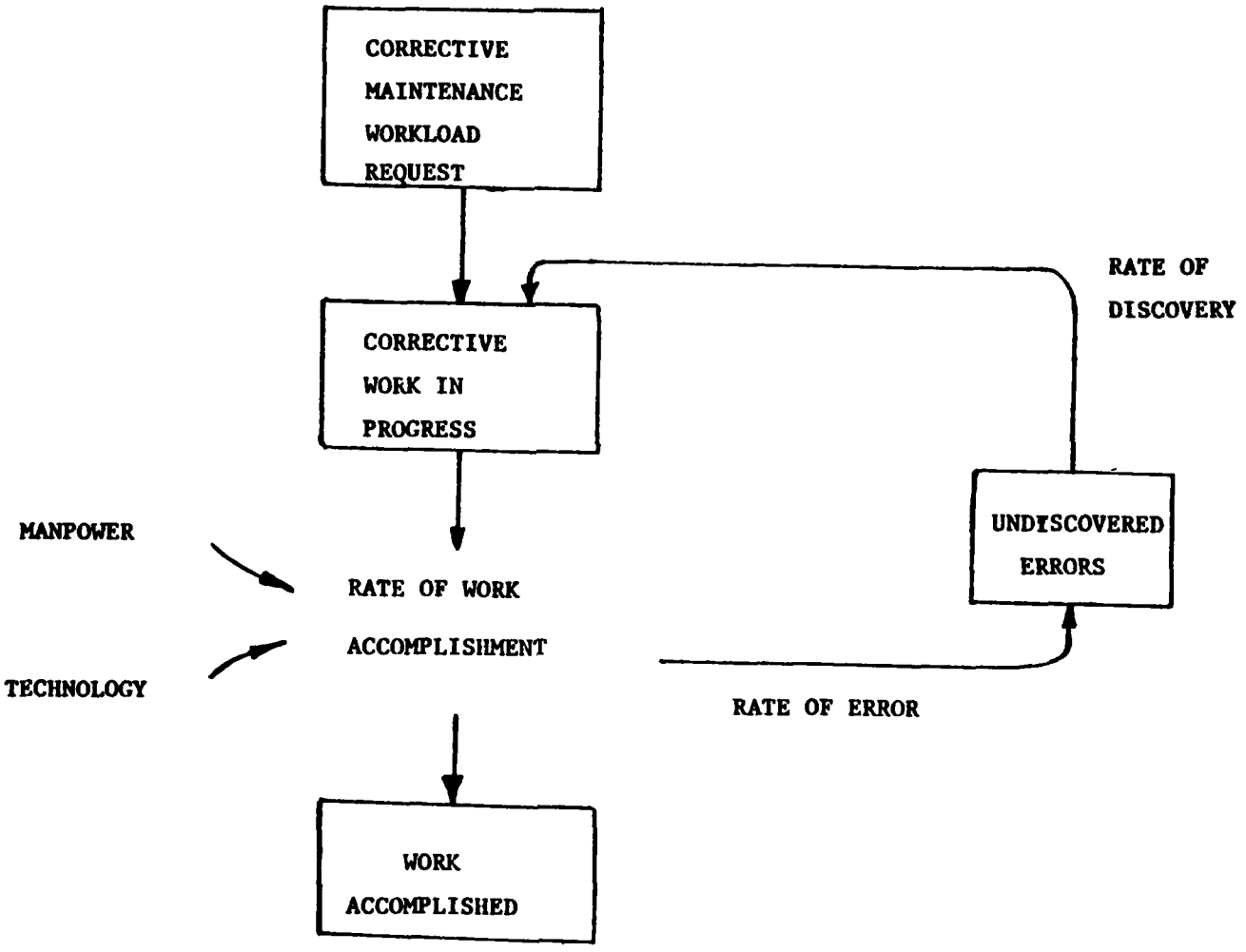


Figure 2-8. Effect of Undiscovered Errors

Chapter Summary

The method of developing and maintaining software is undergoing changes with the objective of reducing life cycle costs. Management controls and programming methods are constantly being developed in order to produce an initial software product which has fewer hidden errors and which can be easily modified to changes in environment, function, and performance. Several management and programming methods have proven to effect maintainability of software. The contention is that these methods increase maintenance productivity, reduce the commitment of personnel, reduce the occurrence of hidden errors, increase the quality of the software product, and ultimately reduce the costs of software maintenance. The extent to which these management and programming methods are applied to Air Force Weapons System software applications is not known. The study described in Chapter 3 attempts to detail the level of application and attempts to investigate additional methods for increasing productivity and lowering maintenance costs.

III. Methodology

The study of software characteristics and qualities has been influenced by large human variability. The variability of programmer preferences and application requirements has caused difficulties in applying value ratings to specific software practices. Preliminary research has shown that attempts to measure the influences upon software maintainability have had the same difficulties. Because of this situation, this thesis research has taken a very structured approach to the study of software maintenance practices within Air Force systems. Three specific research objectives were formalized and all thesis work was applied to these goals. A set of experimental hypotheses were established to provide a basis for accomplishing each objective. The hypotheses were developed based on the literature research findings presented in Chapter II. A questionnaire was constructed to test the validity and to quantify these hypotheses as they apply to Air Force weapon systems. This questionnaire was used as the basis for personal interviews with key management and programmer personnel within a sampling of Air Force systems. This chapter describes the methodology which was used to plan, administer, and analyze the interview investigation. Chapter IV provides a summary of the interview responses as they apply to the thesis objectives. Chapter V outlines software maintenance recommendations.

Sampling Population Description

To accomplish the thesis objectives, managers and programmers from 13 Air Force weapon system software support efforts were interviewed. These systems were selected through a preliminary filtering process so as to insure that the time spent on the thesis investigation would be productively applied. A large group of systems were initially contacted based on available points of reference supplied by the sponsor of this thesis (AFTEC). Further contacts were developed from discussions of personnel from this original group of systems. The list of possible systems to be studied was limited to those which were acquired, developed, or maintained in accordance with the procedural requirement of the 800 series Air Force regulations. Additionally, in order to have some basis for inquiry, the list of systems was limited to those which had been in the maintenance phase of the software life cycle for at least one year. Another prequalification condition was that key management and programming personnel would be available for personnal interviews and follow up questions. These prerequisites were established in order to increase the probability of obtaining answers to posed questions within the limited research time frame of this thesis.

Further constraints were required to limit the number of systems to be investigated. Three categories of systems were chosen from systems meeting established prerequisites: Communications, Avionics, and Ground Radar. The purpose of

choosing these categories of systems was to gather maintenance information from projects which are managed and supported by different Air Force organizations, and have varied mission priorities, objectives, and restrictions. By investigating Air Force systems in this manner it was hoped that common relationships could be established, and successful management solutions to maintenance difficulties could be identified. Systems within these categories were selected for the purpose of observing representative management and programming methods. For example, most ground radar systems are currently managed by Headquarters Strategic Air Command (HQ SAC). The software maintenance functions of these systems is completed by three different types of organizations: contractors, SAC tenant organizations and HQ SAC. A software system from each of these organizational types was selected for study. Table 2 summarizes the systems that were selected and the organizations which were contacted for this study. Appendix 2 provides a description of the mission and system components of each of these systems.

SYSTEM	CONTACTED ORGANIZATIONS
Ground Radar Systems:	
PARCS	HQ SAC, Raytheon
PAVE PAWS	HQ SAC, 3900 CSS
COBRA DANE	HQ SAC, Raytheon
BMEWS	HQ SAC
FPS-85	HQ SAC, 20 MWS
FSS-7	HQ SAC
CCPDS	HQ SAC

Avionics Systems:	
FB-111A	Sacramento-ALC/MME. General Dynamics, Rockwell
F-111D	Sacramento-ALC/MME, General Dynamics, Rockwell
F-111F	Sacramento-ALC/MME, General Dynamics, Rockwell
Communication Systems:	
AFSATCOM	AFCCPC
IEMATS	AFCCPC
PIDP	AFCCPC

Table 2. Air Force Systems Studies

Appendix 2 also lists the key personnel who were interviewed with respect to each system. Even though only a few names are listed, much additional assistance was needed to gather facts necessary to allow the key personnel to respond to the interview questions. In some cases, other personnel were brought in to explain areas in which they were more technically familiar. The result was the formulation of a single response to each interview inquiry made for each system studied. In the case where two opposing responses were given by different personnel within an organization, the differences were successfully resolved by having each person confront the other's point of view.

Data Collection Method

After the systems to be studied were identified, the personnel responsible for the software maintenance

management of these systems were contacted by phone. An explanation of the research objectives and study scope was explained at this time. An appointment date for a personal interview was also established. A message was sent to each oganization in order to establish a formal committment of personnel and to establish necessary security clearances. A TDY trip to HQ SAC-Offutt AFB NE, SAC ALC/MME-McClellan AFB CA and AFCCPC-Tinker AFB OK was completed in order to conduct interviews of key management and programmer personnel at those locations. Because of remote or distributed locations, the remainder of interviews were conducted by phone.

Maintenance information for each system was gathered by interviewing various personnel using a pre-established set of questions. The questions were developed to verify or redefine conceptual hypotheses relating to the management and support of weapon system software. The hypotheses were chosen as the vehicle by which the thesis objectives could be achieved. These hypothesis were established based on the literature research findings discussed in Chapter II. Tables 3A, 3B, and 3C outline the investigation design structure for each thesis objective.

The interview process itself was conducted by first contacting the manager of the shop responsible for performing maintenance of the system's software. Then the research purpose and objectives were further explained. A general description of the types of inquiries that were to

Table 3 A. System Investigation Design Structure - Objective 1

THESIS OBJECTIVE	HYPOTHESIS	INTERVIEW QUESTIONS
1. Identify the factors which influence the level of maintenance costs.	1. Weapon System Software projects can be categorized as a single application type.	1. Section I Q 4,5,6e,7 Section II Q 1,2,4-7, 18
	2. The current level of maintenance costs can be estimated.	2. Section II Q 5,6,8, 12,13,14,15
	3. The value of the following variables is a function of system age: a. Size of program b. Man years dedicated to maintenance c. Software Reliability	3. Section II Q 4-7
	4. There is a relationship between the perceptions of the current state of several factors and the level of maintenance effort.	4. Section II Q 4-7,9-15, 18 Section III Q 3

Table 3 B. System Investigation Design Structure - Objective 2

THESIS OBJECTIVE	HYPOTHESIS	INTERVIEW QUESTIONS
2. To identify Air Force policies which tend to cause excess resource commitments to the software maintenance of weapon systems.	1. Software maintenance is defined differently by different organizations.	1. Section II Q 1,2
	2. During system development, software quality is not a primary consideration.	2. Section I 6f-h,7
	3. There does not exists standard maintenance programming methods, procedures, or goals.	3. Section II Q 17,18, 18a
	4. Personnel assigned to software maintenance tend to be insufficiently trained and have limited software support experience.	4. Section I Q 6f Section II Q 3,11 Section III Q 1b
	5. There is a common set of problems which plague the maintenance life cycle of Air Force Weapon System software.	5. Section III Q 1,2

Table 3 C. System Investigation Design Structure - Objective 3

THESIS OBJECTIVE	HYPOTHESIS	INTERVIEW QUESTIONS
3. To develop policy recommendations to im maintenance support of existing and future software programs.	1. Steps are being taken to improve the maintenance of weapon system software.	1. Background investigation
	2. Analyzing maintenance activity data would assist with managing the maintenance process.	2. Section IV Q 1-7
	3. Current management and programmer personnel have viable suggestions for improving the maintenance of existing systems.	3. Section II Q 16 Section III Q 4-7

be made was also given at this time. Then a time was scheduled for a meeting with the personnel best cabable of answering the inquires. In most cases, these personnel included the manager and senior members of the staff. While waiting for this meeting, key system documentation was obtained and studied. The following documents were reviewed:

1. Computer Resources Integrated Support Plan (CRISP)
2. Software Configuration Management Plan
3. Software Programmers Users Manual
4. Local Operating Instructions, Regulations, Standards, and Procedures
5. Recent Software Maintenance Related Correspondence

From these documents, many of the questions in the pre-established questionnaire were answered. In addition, a general understanding of the system's mission and standard operating procedures was obtained. As a result, the actual structured interview process involved the clarification of information found in the reviewed documentation and answering questions not answered in the questionnaire. In some instances, another organization was better qualified to answer some of the inquiries. In these cases, points of contact were established and these questions were answered at a later time.

Interview Questionnaire Description

Table 3A diagrams the investigation structure for the first thesis objective. The hypotheses for this objective were chosen in order to define a relationship between the commitment of organizational resources and the level of

software maintenance workload that is accomplished. The questions used in conjunction with these hypotheses were aimed at gathering expert opinions and judgements as to the extent these relationships hold true.

Table 3B outlines the method for accomplishing the second thesis objective. The hypotheses supporting the second objective were chosen in order to produce a clear picture of how current Air Force policies affect the balance of the relationships established in the first thesis objective. This was accomplished by asking questions aimed at identifying areas which cause hardships for maintenance managers and programmers. Once these areas were identified, then the policies which create these conditions could be investigated.

Table 3C identifies the approach used for developing some useful recommendations for improving the maintenance process (i.e. thesis objective 3). Generally, these recommendations were developed based on the findings of the previous objectives. Yet, before these findings could be applied, further study was needed. The first hypothesis investigated what methods have been tried or are under development. The second hypothesis investigated the application of a maintenance improvement recommendation made by several previous studies. The third hypothesis attempted to solicit any untried improvement ideas from current personnel. The specific findings of these three hypotheses are discussed in Chapter IV. A total list of

recommendations formulated to complete the third thesis objective is presented in Chapter V.

As mentioned, each hypothesis has an associated group of questions which are used to validate or negate the hypothesis statement. The content of each question was developed to further the investigation of the hypotheses.

Some questions are used in conjunction with several hypothesis. Using this structure, each question that is asked during the interview is directly related to thesis objectives.

The organization of the questions is based on a logical interview presentation. The terminology and question format were modeled after the questions used in previous studies (Ref 52). The questions were designed to collect both factual and perceived information. The questions were partitioned into four sections:

Section 1: General System Description

Section 2: Maintenance Effort - Facts

Section 3: Maintenance Effort - Perceptions

Section 4: Data Measurement Evaluation

Questions in the first section are designed to identify the weapon system and to describe the development of the software product. The next two sections attempt to define the maintenance life cycle, software growth trends, and the management/programming methods used to support software. One section identifies factual information; the other section attempts to draw on the opinions and experience of

software maintenance personnel. Section 4 investigates the collection of maintenance activity data.

Data Analysis Approach

The results of the interviews are presented in Chapter IV through a discussion of each objective hypothesis. Depending on the type of question used to investigate the hypothesis, the interview responses are summarized in several different ways. Responses are presented through narative, pie charts, causal-loop diagrams and other graphical methods. These summaries are based on a reivew of the responses and the application of simple non parametric statistical analysis. Appendix 3 details the three major statistical data analysis methods used. Further statistical analysis was not warranted because of the limited sample size and the general nature of the data being sampled. To support the statistical conclusions made in Chapter IV, Appendix 4 contains a list of responses to those questions which were analyzed statistically. Appendix 4 also contains the appropriate data analysis solutions to the recorded responses.

After each hypothesis is discussed, the interview findings are applied to directly answering the thesis objective. This process is completed for each thesis objective in the next chapter.

IV. Data Analysis of Software Maintenance Investigation

This chapter presents an analysis of the responses which were gathered during structured interviews of key personnel involved with the software maintenance of weapon system computers. The interview responses provided a descriptive picture of the current state of software support conditions within observed systems. In which software support conditions are constantly changing. The analysis of the interview data served as a basis for validating a set of pre-established hypotheses and for accomplishing each thesis objective. Supporting hypotheses for each thesis objective examined and the results of each hypothesis test summarize and relate to accomplishing each thesis objective.

OBJECTIVE 1: To Identify The Factors Which Influence The Level of Maintenance Costs Within Air Force Weapon Systems

HYPOTHESIS 1 - Weapon System software projects can be categorized as a single application type.

This hypothesis was designed to establish the characteristics of the group of systems that were studied. The discussion of these characteristics of this class of computer applications provides the basis by which the conclusions drawn from this study can be applied to other weapon systems.

The software projects that were investigated have displayed similarities and differences. Several differences were observed between the systems studied:

- Management Organization
- Maintenance Organization
- Computer Hardware
- Software Languages
- Unique Mission Requirements

Each of the three groups of weapon systems is managed by a different functional organization:

Ground Radar Systems	- Headquarters Strategic Air Command (HQ SAC)
Communications	- Air Force Communications Command (AFCC)
Avionics (F-111)	- Air Force Logistics Command (AFLC) through Sacramento Air Logistics Center (SAC ALC)

Each functional manager has a different philosophy concerning how software maintenance functions are to be handled. AFCC provides software maintenance functions through a centralized software development and test center. This facility shares management and technical expertice for performing software maintenance on communications computer systems. HQ SAC has three approaches to accomplishing software maintenance. The first approach is to embed software maintenance tasks into a contract for complete system operation and maintenance. The second approach is for HQ SAC to perform the software maintenance tasks internally. The third approach is to formulated a tenant or support organization to perform software and hardware support functions. AFLC delegates system support of weapon systems to one of the Air Logistic Centers (ALC). Within each ALC, the engineering division is responsible for supporting software maintenance functions. This support may

be in the form of an inhouse or contractor provided work force. As a result of the differing management and workforce organizational structures, there are several different approachs taken towards to managing and performing maintenance functions. These variations were observed by noting differences in local interpretations of Air Force regulations and differences in contract software maintenance tasks required of contractors.

Such variations in the management of software support appear to be due to the unique and specialized missions of each weapon system. The specialized missions require unique and sometimes one-of-a-kind computer systems to integrate with one-of-a-kind weapon systems. From the systems observed, a vast range of computers were being utilized. Large main frames such as CDC CYBERs, and IBM 360/370's are being used to support ground radar systems. Mid-size machines such as Burroughs and Data General Nova are used for communications applications. Special purpose AN/AVK-6 are used to support general navigation and weapons delivery within avionics weapon systems. The presence of different computer systems has caused a proliferation of programming languages used to build the software. Within the thirteen systems studied, over eight different languages were being utilized.

The specialized mission requirements require different levels of software function complexity. For example, the

FB-111 navigational computer performs a limited set of navigational operations using approximately 250 instructions; where as, the PAVE PAWS phased array warning system utilizes over one million lines of code to provide SLBM warning and attack characterization.

Many of the wide variances that were found from this study are a result of the research philosophy used for choosing the weapon systems to be studied. There was an intentional attempt to look at a representative sample of systems so that a comprehensive perspective of Air Force software maintenance could be attained.

Even though there were many differences between systems, there were also numerous similarities. For example, computer systems are consistently used to fulfill the same weapon system function, that is, to act as the brain or controller of the hardware.

Additionally, the computer resources have been acquired in the same manner, that is, as a component of the weapon system acquisition. As a result, the software has been developed by a contractor under Air Force direction and management. This development typically undergoes the standard Air Force configuration management, verification and validation, and operational test, evaluation and acceptance phases. Software life cycle planning is also accomplished using a common set of regulations and procedures.

Each of the weapon systems that were studied, required the computer resources to provide real time data processing. As will be discussed in greater detail later. this has caused each computer application to be faced with high speed response requirements. memory restrictions, and reliability concerns. To meet the speed and efficiency requirements, assembly languages are often used to develop the software. Of the software applications investigated, assembly languages are used a high percentage of the time. as shown below:

Assembly	-	62 percent
Jovial	-	12 percent
Fortran	-	10 percent
Unique Languages	-	18 percent

Common processing requirements are shared by computer resources because software performs a critical role in supporting the weapon system.

As also found by the BDM study (Ref), the software maintenance environment of each system studied has a similar structure. Management methods and programmer tools are used to complete maintenance tasks as outlined in Figure 2-9. This is in a large part due to Air Force and DOD policies, procedures and standards which outline configuration management and software development requirements.

Because of the diverse characteristics of each weapon system, it is sometimes difficult to even categorize the studied systems into the general categories of avionics.

communications, and ground radar. Yet, based on common development, processing, operations and maintenance philosophies, there is rationale for categorizing weapon system software as a single application. This evaluation becomes increasingly clear when weapon system software projects are compared with other Air Force data processing applications and especially true when compared to business and scientific processing of non-defense related applications. Real time processing requirements, use of assembly languages, software reliability concerns, and similar software support facilities give weapon system software applications commonality.

HYPOTHESIS 2 - The current level of maintenance costs can be estimated

As discussed earlier, software maintenance tends to be controlled as a function. This entails the dedication of resources in the form of personnel and computer components to complete maintenance tasks. The resources are assigned to the organization which has the software support responsibility. Accounting for these resources is done on a limited basis. When the responsible organization is a contractor, resource commitment accounting can be found by inspecting the contract statement of work (SOW) or the contractor's proposal. Depending on the type of contract which is used, further resource costs can be estimated through periodic work break down status reports. One problem that was encountered in estimating actual resource

expenditures to contractor maintenance, was breaking out software versus system maintenance resource expenditures. The majority of contracted software maintenance efforts are embedded within a total system maintenance contracts. So in order to win the competitive bid of system maintenance, contractors will utilize personnel and other resources to complete both system and software maintenance tasks making it difficult for Air Force managers to identify actual software maintenance costs. Thus, total software maintenance costs were not readily available for this study. Estimations of costs were made by reviewing the available contracts, and expense reports.

When the responsible maintenance organization was the Air Force, resource expenditures could be identified with less difficulty. The cost of software maintenance-support computer resources was found by reviewing purchase or lease agreements. In a few cases this approach was not feasible. On some systems, software maintenance is not performed on off-line computer rosources; maintenance is performed on the target machine during non critical time periods. Personnel costs were estimated using current manning multiplied times current salaries. Administrative, support and travel costs were obtained from the current fiscal year budget. Even though these expenses could be roughly estimated through this study, there appeared to be no attempt to collectively monitor and control these expenditures.

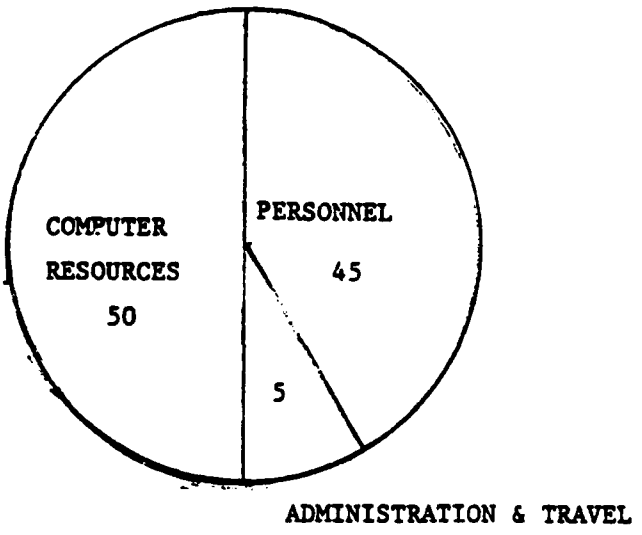


Figure 4-1. The Software Maintenance Dollar

The Figure 4-1 summarizes the way resources are committed to the software maintenance function. Maintenance costs estimates were ranked in order of magnitude. Even though the size of resource expenditures varied, there was relative agreement as to the ranking of the categories of maintenance costs.[1] The cost relationships held true for systems requiring 5 to 50 personnel to complete software

[1] Rank association was verified using the Kendall coefficient of concordance as demonstrated in Appendix C (The Chi Square Distribution Table was used to establish the P value to be greater than .1).

maintenance functions. The largest variances were seen with systems that are in remote locations. In these cases, personnel and travel costs tended to be higher. With systems requiring fewer structure maintainers, computer resources costs tended to be the dominant factor.

HYPOTHESIS 3 - The value of the following variables is a function of system age:

a. Size of program

b. Software reliability

c. Man years of maintenance workload

When the majority of systems are deployed, memory is utilized at 90-95 percent because of the influence of acquisition on system design. The computer product is designed and developed to meet the current functional specification only. To minimize system costs. software is developed to maximize the usage of hardware. Thus, when systems enter the maintenance phase there is little room for software growth. At this point, all software changes must be carefully considered. Before any significant new software additions can be made, some portion of the current software baseline must be deleted. Most software projects show about a 5 to 10 percent exchange of program code during the scheduled version change. This statistic more accurately measures the growth of software with respect to time.

When the system goes operational, the software is subjected to a large array of conditions not tested during

development. Because of this situation, a relatively large number of errors are detected during this time period. As these errors are corrected, the system establishes stability and the software gains greater reliability. It is the general opinion of software maintainers that software reliability increases the longer the system is operational.

From the date that the weapon system is initially deployed, performance requirements of the system begin to change. As the weapon system ages, modifications are made to keep up with these requirement changes. Sometimes these system modifications take the form of software function enhancements. Eventually, the system can no longer be altered to meet mission requirements. and is replaced by a new system. As the system approaches replacement, both the hardware and software configurations are held constant; Changes are no longer cost justified. Figure 4-3 presents the expenditure of man hours to support software changes over the system life span. The initial expenditure of man hours is dedicated to

1) handling the transition of software from development,

2) establishing the software maintenance facility and management procedures, and

3) correcting detected errors during initial deployment.

The expenditure of software maintenance man hours levels as the software and system stabilizes. As the reliability and

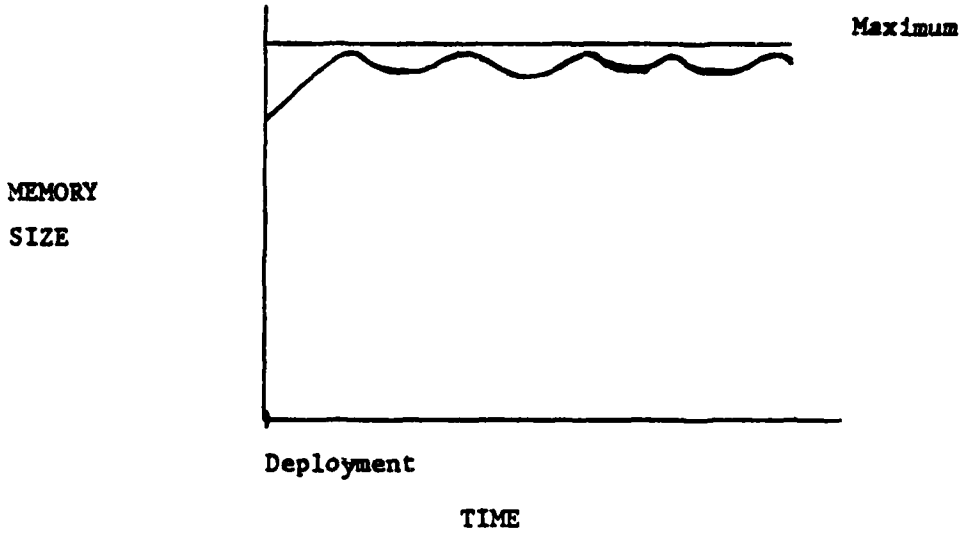


Figure 4-2. Software Growth Over Time

Figure 4-2 demonstrates this reported growth trend over time.

quality of the software increases, the number of man hours gradually decreases until software changes are discontinued and the system is frozen. Both the BMEWS and AN/FSS-7 systems are currently frozen.

HYPOTHESIS 4 - There is a relationship between the perceptions of the current state of several factors and the level of maintenance effort.

Figure 2-9 presents the management and programmer influences on the maintenance process. A series of questions were designed to investiate the factors which affect the number of maintenance tasks which are identified and the amount of time needed to complete those tasks.

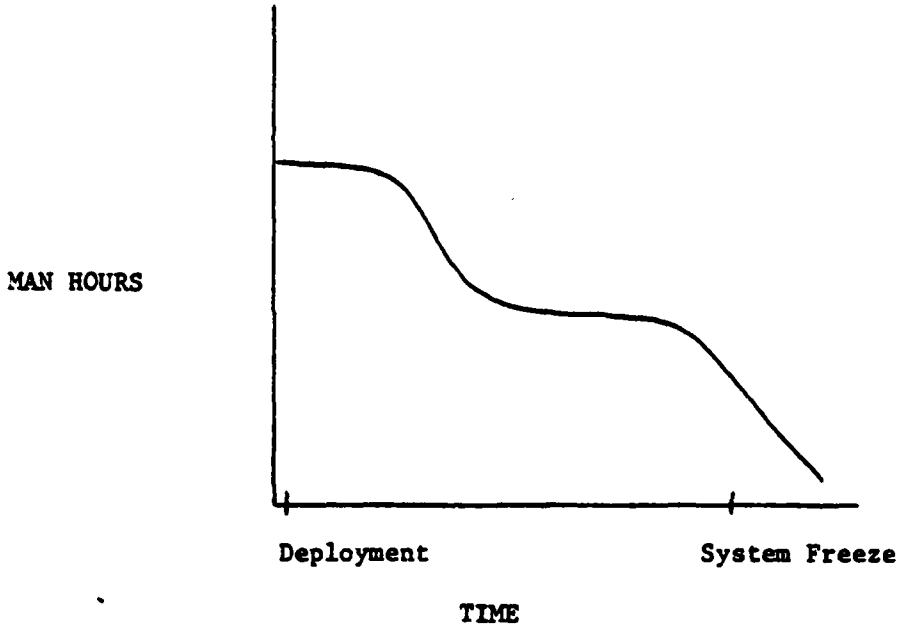


Figure 4-3. Level of Maintenance Workload by Time

Responses to Section III, Question 3 of the questionnaire (Appendix 1) have identified four major factors which influence the level of maintenance effort:

1. System age
2. Code and documentation quality
3. Experience of maintenance programmers
4. Level of user enhancements

These factors were identified by asking software maintainers to rank the factors which most affect the level of software maintenance workload. Analysis of responses (Refer to Appendix C-Problem 2) pointed to the factors listed above. Additional questions were utilized to describe the relationship between these factors and maintenance level of effort. Figure 4-4 illustrates the

causal relationships which were established as a result of the further questioning. The answers to the questions supporting Hypothesis 3 established that maintenance man hours tend to decrease as the system gets older. Related results explained the role that software reliability has in affecting the maintenance workload. Reliability is only one software quality measure which affects maintenance. Indeed, from the viewpoint of software maintainers, reliability is the most significant measure of program code quality. The quality of documentation can influence the amount of time programmers need to understand program code, analyze solution alternatives, and implement required code changes or additions. As these program quality measures are improved the amount of time needed to accomplish maintenance tasks is diminished.

Software support personnel generally agree that knowledge and experience with the weapon system, the software maintenance facility, the application software code, and local standards and procedures greatly increases the productivity of maintenance managers and programmers. A knowledgeable and experienced person can accomplish assigned tasks in a shorter time period. As knowledge and experience are gained, the significance of this factor is reduced. For example, a programmer with ten years of system software experience would not have a significant advantage over one having eight years of experience. Yet a programmer with two years experience would typically be much more productive

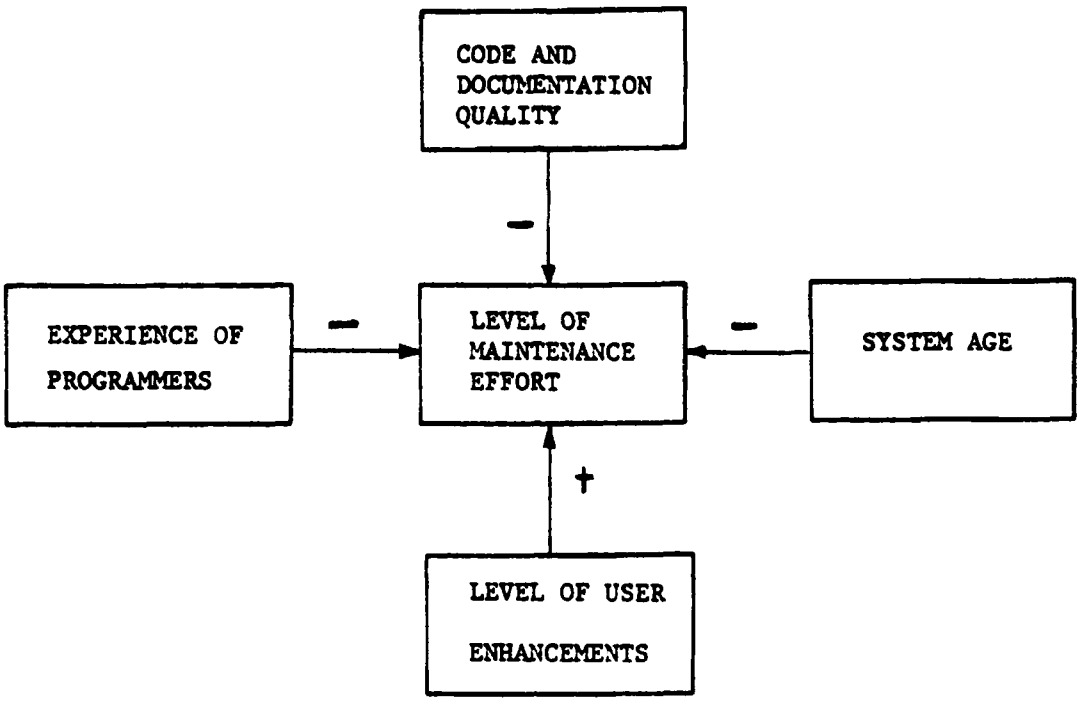


Figure 4-4. Factors Affecting the Maintenance Workload

than one having six months experience.

Of the four factors identified, the number of user enhancement requests appeared to influence the maintenance workload the greatest. Indeed, backlogs of user enhancement requests are often used as the basis for identifying software maintenance deliverables in the contract statement of work or for justifying additional personnel and computer resources. Even though user enhancement requests are a major cause for maintenance tasks, only a limited number of these tasks can be completed because of system and organizational constraints. As detailed earlier, computer

memory limits the number of additions that can be implemented to only those that can replace lower priority functions. The number of available software maintainers also limits the amount of work that can be accomplished. As a system enters the operational and maintenance phase, management establishes a shop of personnel that will provide software maintenance functions. At this time management makes resource commitments to maintenance based on their best predictions and judgements. The level of these commitments then varies as the system evolves. When a user requests software enhancements, these tasks must be handled by available resources. As a result, management verifies and prioritizes change requests. The software support shop then takes a group of the highest priority enhancements and incorporates these requests into a software baseline version change. This causes a certain number of requests to go unanswered until either the priority is large enough or resources are available. If the backlog of high priority requests becomes too large then management attempts to increase the number of programmer personnel or acquires additional automated tools. But both these management efforts have limitations. For as Brooks reports in the Mythical Man Month, adding resources to software development project does not always increase productivity. For one thing, the new personnel would be inexperienced with the system and of little immediate assistance. In fact, training these new personnel would take away from the

current experienced work force. Additionally, software maintenance managers agree that only a limited number of changes can be made within a software version change. Too many changes tend to increase the occurrence of detected and undetected errors. Because of these limitations, management usually bites the bullet and lives with request backlogs. Thus software maintenance personnel manning levels usually stay consistent after the system and software stabilizes.

OBJECTIVE 1 Summary

The responses from the interviews have established a basis for defining the relationships between a few key factors and the level of maintenance costs within Air Force systems. It was discovered that maintenance costs are based on the level of resources that are committed to maintenance tasks. (Refer to Figure 4-4) The commitment of resources is influenced by four major factors: experience of programmers, system age, documentation and program code quality and the level of user enhancements. Interviewed personnel agreed that these factors either affect the amount of maintenance workload or the ability to complete the workload. It was found that the strength of these relationships vary over time. That is to say, the further the code and documentation quality is diminished, the greater the amount of work that must be expended to complete maintenance functions. The same situation is true of system age and the experience of programmers. As the number of use

enhancements increases so does the level of the maintenance workload.

OBJECTIVE 2 To Identify Air Force Policies Which Tend To Cause Excess Resource Commitments To Software Maintenance Of Weapon Systems

HYPOTHESIS 1 - Software maintenance is defined differently by different organizations

Of all the systems that were investigated during this study, all but one agreed on the functions that are performed during the software maintenance process. In fact these systems tended to similarily rank the functions in relation to the total number of maintenance tasks.[2] The responses to Section II, Question 2 are summarized below:

Ranking	Maintenance Function	Mean Percentage
1	Providing new software functions	39
2	Removing software defects	28
3	Tuning the software for greater efficiency	17
4	Providing greater system interfaces	11
5	Other functions	5

Providing new software enhancement functions is seen to be the most significant maintenance activity. During the

[2]The Kendall coefficent of concordance was utilized in conjunction with a chi square significance test to evaluate rank association of responses to Section II, Question 2 (as shown in problem 1 - Appendix C).

questioning, an additional maintenance function was identified. Software maintainers periodically spend time investigating and analyzing reported error conditions that were not actual error conditions. Handling these false "wolf reports" is seen as a necessary service in order to insure a high degree of software credibility.

HYPOTHESIS 2 - During system development, software quality is not a primary consideration

When trying to identify the critical measures of software quality, software maintainers tend to agree that software reliability is the most visible. Software error occurrences often affect the entire weapon system performance. For this reason, a significant amount of time and resources are expended on software test and integration during the software maintenance phase. In general, software maintenance personnel assert that this same concern for software reliability is usually not adequately considered during the initial software development. Indeed, several software development experiences tend to support this assertion. For example, the software for IEMATS was allowed to be developed in a "hodge-podge" manner. The contractor was permitted to integrate several software modules from other systems in order to reduce software development costs. This resulted in a software product which inherited the combined problems of the previous systems. Inadequate software documentation, and poorly structured program code

were only two of the problems which later greatly affected software maintenance personnel. Another case in point was the development of the Programmable Indicator Data Processor (PIDP) communications system. Because of separate developments, the software was delivered several months before the hardware was available. Thus, there was limited means by which the software could be adequately operationally tested. So when the hardware was finally delivered, the software developer had long been paid off. When the Air Force attempted to integrate and run the hardware/software system, it was found that the software did not execute. And even when fixed to do so, the software did not perform the functions specified in the software development contact. In this case, the software maintenance team was given the responsibility of reworking the software to meet the original systems functional specification.

The role of correcting problems created during the development phase is not an uncommon one. Software support personnel consistently regard the modified program code as of being of better quality. One reason for this situation, is that the code is developed consistent with local standards using existing maintenance programming tools. The experience gained working with the new section of program code coupled with the enforcement of local maintenance standards tends to make the code more maintainable later in the life cycle.

This study found that approximately 30 percent of software maintenance activity involved spent correcting errors created earlier in the life cycle. The rising costs of performing maintenance justifies reconsideration of several software development policies and procedures. Possibly by spending additional resources during system development, a higher degree of software reliability and maintainability could be attained.

HYPOTHESIS 3 - There does not exist standard maintenance programming methods, procedures, or goals

There are a series of DOD and Air Force regulations and standards which are directed at controlling the state of delivered software products. Areas which are specifically identified and controlled during the software development phase are configuration management and system/software documentation requirements. The same level of management policy control has not been developed for the maintenance phase of the software life cycle. Section II, Question 15 and 16 were directed at discovering the standards and procedures for performing code modifications. Responses to these questions indicated a wide diversification of methods which are used by the software maintainers of different weapon systems. Some systems tended to utilize Air Force software development standards; other systems emphasized the use modern methods of software development and testing. The Air Force Communications Command (AFCC) has realized this

situation and is attempting to reduce the impact by centralizing the software maintenance efforts of communications systems. Preliminary work has been done to manage and program these systems using a common approach. Yet, further standards are needed.

HQ SAC is also attempting to manage this predicament. HQ SAC has recognized that some software systems are deficient in many quality attributes. Yet, managers are not sure what is an acceptable level or standard by which software should be maintained. In attempt to answer this question, HQ SAC is developing an Request For Proposal (RFP) to have a contractor study a model software system. The purpose of the study will be to develop acceptable documentation and program coding standards for maintaining software. The contractor will also estimate the costs of converting the software from its current state to that of the established standards. Current conversion estimates are high.

HYPOTHESIS 4 - Personnel assigned to software maintenance tend to be insufficiently trained and have limited software support experience

Depending on the weapon system, either contractor or Air Force personnel are responsible for completing software maintenance tasks. Personnel from both types of organizations agree that it takes a special combination of knowledge and experience to make an effective software maintenance programmer/analyst. As Figure 4-5 illustrates.

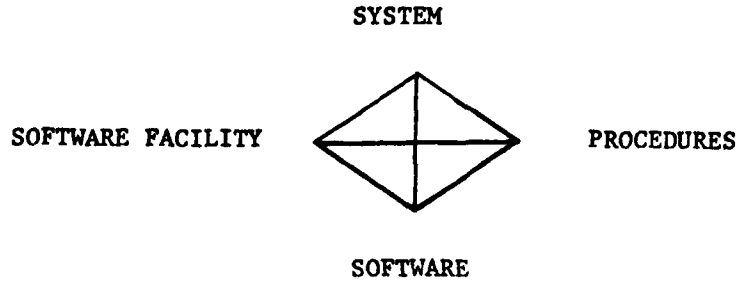


Figure 4-5. Software Maintenance Expertise Areas

the programmer/analyst must obtain expertise in four areas: the weapon system, software application, the software support facility, and local standards and procedures for completing maintenance tasks. System knowledge is often referred to as knowledge of the software application environment. By acquiring at least general knowledge of the system. the software maintainer will develop an understanding for system requirements and will be more able to interface with change requests to assure that requirements are sufficiently met. When the software support programmer goes through the process of making initial changes to the software much valuable experience is

gained. The programmer is subject to working with the internal and external software documentation, representative program logic and the programming language. As the programmer completes maintenance tasks, a point of reference and familiarity is developed. As a result, the programmer can anlyze software problems and solutions having some basis for working with the software. It's one thing to know what changes need to be made, another to be able to develop and test them. Experience using the software maintenance automated tools gives the programmer the ability to accomplish code modifications. Just as a carpenter requires special knowledge of each of his tools; software maintenance personnel need to be able to operate the text editors, debugging packages, dump routines, assemblers and compilers and other software packages available in the software support facility. Knowledge of local standards and procedures allows the programmer to apply the expertise gained in the other areas to completing assigned maintenance tasks. It takes learning a combination of these four areas, to develop an effective software maintenance programmer/analyst.

Expertise with these four areas is usually developed through an inhouse training program coupled with hands-on experience. All the systems investigated had a thorough training program defined for developing newly assigned personnel. The major purpose of the training is to gradually impart system and software knowledge and at the

same time to enforce acceptable means for accomplishing software maintenance tasks. After the new personnel complete the training program. they are usually assigned to be an apprentice of an experience programmer/analyst. The new programmer is assigned small tasks and directed to shadow the actions of the experienced individual. Eventually. the new programmer is given tasks with increasing importance and with decreasing guidance. The personnel within each system agreed that the process of grooming experienced software maintenance programmer/analysts takes between 8 to 18 months to complete. The exact development time depends on educational background, job experience and aptitude to learn applications and software concepts. Because of the diversity of systems, much of the learning necessary can only be accomplished on the job.

Managers of the systems interviewed believe that quality, experienced personnel are the key to the completing software maintenance tasks. Achieving this experience is a difficult and expensive undertaking. The personnel within Air Force software maintenance shops average less than 2 years experience working on any software support project. For most personnel. their current assignment is their only experience with software support functions. Turnover of management and programmer personnel is estimated to be just over 3 years. On the other hand, contractor personnel experience averages almost three times that of the Air

Force. As a result, the Air Force pays 65 thousand a year for the average contractor software maintenance individual. This fee is much higher for remote software support sites.

Current Air Force policy does not enable Air Force software maintenance shops to keep needed experienced personnel. Obtaining the needed level of software support experience takes longer than most other applications. Just as maintenance personnel have gained the management and programming skills equivalent to their contractor contemporaries, they are transferred or promoted. The time and expense of constantly training new personnel, detracts from the amount of software maintenance work that can be accomplished and drives up the cost of supporting the software through the life cycle. The loss of experience is particularly acute at management levels. For example, in the case of contracting software maintenance tasks, Air Force contractor negotiations tend to be one sided in favor of the contractor. The contractor holds the knowledge, experience and skills and can strongly influence the identification and pricing of maintenance tasks. This situation is referred to as major league-minor league contract negotiations. The Air Force personnel do not have the experience or technical knowledge necessary to keep the contractor in check. The hard work and perseverance of the Air Force may win some concessions but the contractor wins overall. Once again, level of personnel experience influences the price tag of software support.

HYPOTHESIS 5 - There is a common set of problem conditions which plague the software life cycle of Air Force weapon systems

The hypotheses that were developed under this objective were aimed at identifying specific conditions which cause unnecessary expenditure of resources to maintenance activities. The purpose of these hypothesis was to investigate the extent to which potential problem areas apply to the different weapon systems studied. Table 4 summarizes the responses to Section III, Question 2. This question listed all the potential problem conditions reported in the software maintenance of business applications (Ref 52). The table shows the range of responses by indicating the maximum, minimum and medium values. The purpose of this question was to gain a comprehensive viewpoint of which situations cause maintenance management and programming difficulties. Responses that had a median value of minor problem (score value 3) or higher were categorized as a problem area. Six problem areas were identified:

1. Turnover of maintenance personnel
2. Quality of application/system documentation
3. User demand for enhancements and extensions
4. Quality of original program development
5. Lack of user understanding of application
6. Storage requirements of system programs

Table 4. Problem Area Response Description

	None at All	Somewhat Minor	Minor Problem	Somewhat Major	Major Problem
	5	4	3	2	1
a. Turnover of maintenance personnel					
b. Quality of application system documentation					
c. Changes made to system hardware and software					
d. User demand for enhancements and extensions to application systems					
e. Skills of maintenance programming personnel					
f. Quality of original programming of application system					
g. Number of maintenance programming personnel available					
h. Competing demands for maintenance programming personnel time					
i. Lack of user interest in application system					
j. Application system run failures					
k. Lack of user understanding or application system					

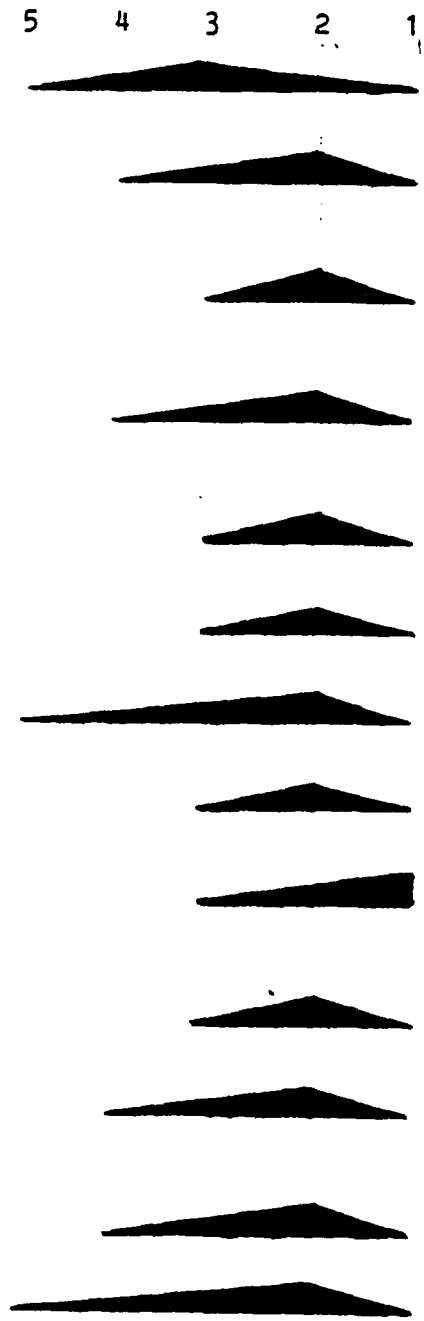
None at All

Somewhat Minor

Minor Problem

Somewhat Major

Major Problem



l. Storage requirements of application system programs

m. Processing time requirements of application system programs

n. Motivation of maintenance programming personnel

o. Forecasting of maintenance programming personnel requirements

p. Maintenance programming productivity

q. System hardware and software reliability

r. Data integrity in application system

s. Unrealistic user expectations

t. Adherence to programming standards in maintenance

u. Management support of application system

v. Adequacy of application system design specifications

w. Budgetary pressures

x. Meeting scheduled committments

The impact of each of these problem conditions has been discussed in detail within an earlier hypothesis. The recommendations offered in Chapter V will focus in on reducing the impact of these conditions on software maintenance costs.

OBJECTIVE 2 Summmary

This section has identified a series of Air Force policies which cause excess resources to be committed to software maintenance tasks. The turnover rate of Air Force personnel causes a constant deficiency of experienced managers and programmers. In the past. the system development phase has produced software which has a low level of software reliability. Software is often delivered without the needed documentation to support the program code. Lack of maintenance standards creates difficulties for the transition of software from development to the maintenance phase. For if software maintainers cannot agree on the level of documentation, and code quality needed then software developers will have a hard time delivering an acceptable software product. The distributed approach to organizing the software maintenance function within the operational-using command causes the impact of these policies to be greater. Personnel expertise is isolated on separate projects. Different organizations tend to support different philosophies for completing software maintenance.

The Air Force has made incredible progress with controlling the explosive emergence of software support as a major management concern. The third thesis objective will investigate the progress that has been made to date.

OBJECTIVE 3 - To Develop Policy Recommendations To Improve Maintenance Support Of Existing and Future Software Programs

HYPOTHESIS 1 - Steps are being taken to improve the maintenance of weapon system software

Several positive steps have already been made to improve software maintenance with Air Force weapon systems; Several new developments are trying to improve the process even more. The Air Force has long realized the importance of having a total systems approach to maintenance. The functional manager recognizes that a well maintained weapon system is the product of an integrated process of hardware. software and the personnel and equipment to support both. As part of this systems approach, software maintenance is organized within a distinct and separate organization. The approach of assigning a separate organization to handle maintenance responsibilities has been recommended by several studies as a method to increase personnel and resource productivity (Refer).

Detailed configuration management programs involve Air Force decision makers with authorizing and validating the maintenance tasks which are to be accomplished. This control coupled with scheduled implementation of authorized

changes has stopped the "quick change" syndrome of accomplishing software maintenance. As a result, software users, managers and programmers are sure of the current software configuration. Software reliability is better, also.

Automated tools are being used by programmers and managers to accomplish software maintenance functions that have long been done manually. The Air Force is utilizing these tools in increasing degree. The current trend is to create a software maintenance facility or environment to support weapon system software. The philosophy of these new maintenance facilities is to integrate a comprehensive set of tools with management and programming methods. Science Applications Inc. has developed a metric-directed maintenance environment using this philosophy.

New research and development projects promise to further assist managers and programmers with maintenance tasks. Development of the ADA programming language and the ADA development environment will provide a common software tool for the various Air Force weapon systems. As mentioned earlier, the need for efficient real time processing on different specialized computers, has caused a wide variance of programming languages. This diversity is one of the road blocks stopping further advances in programmer productivity. Introductin to ADA a common programming language will provide a framework for developing standard programming conventions and methods. DOD and the Air Force have high

expectations for this avenue of work.

Rome Air Development Center (RADC) has sponsored several software quality metrics research efforts The purpose of this research is to identify measurable software qualities which can be attained during the development of software. McCall and Walters (Ref 62 & 63) have defined a set of software maintainability metrics. These metrics describe software attributes which allow software to be corrected and updating much easier. There are currently plans to require contractors to attain these maintainability attributes during software development. Future use of these concepts may insure a higher quality software product.

On a large scale, revolutionary new methods of developing and maintaining software are being researched. Some of the major efforts are:

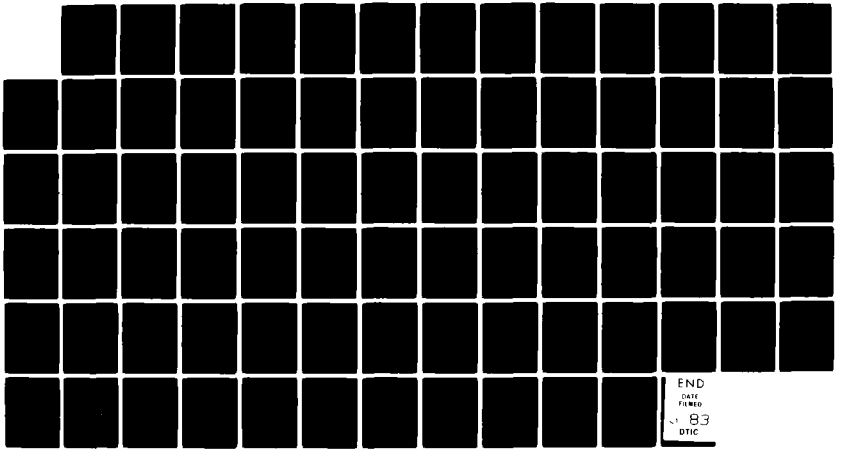
- Requirement Specification Languages
- Program Correctness Proofs
- Automatic Programming

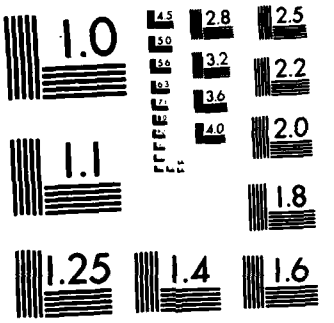
Requirements specification language is a short hand method of stating a user requirement (Ref 7,84,85 & 90). The language can be directly interpreted into executable code. As user requirements change, the requirements language description can be modified, thus changing the executable code. By using this type of tool, "adaptive" type maintenance efforts could be automated. Much work is needed to apply current requirements languages to the efficiency needs of weapon system software.

AD-A124 758 A STUDY OF THE SOFTWARE MAINTENANCE PROCESS OF AIR FORCE WEAPON SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING

UNCLASSIFIED J P JOYCE DEC 82 AFIT/GCS/MA/82D-5 F/G 9/2 NL

2/2





Program correctness proofs are a mathmatical method of proving program validity (Ref 38 & 41). Correctness proofs could be used during the design and coding stages of development to reduce the occurence of errors. This approach promises to minimize the amount of "corrective" maintenance tasks. Work is underway to apply current correctness proof concepts to complex programs. Further development of these concepts is required before it becomes cost and time effective to implement these methods.

Automatic programming is a programming method which allows the computer to program all low level tasks (Ref 5,33,37 & 103). This approach leaves the programmer to work at a very high level of program specification. If properly developed, this programming method could automatically implement efficient well tuned code for all low level tasks, thus, greatly reducing the level of perfective maintenance tasks. Prototype automatic programming languages have been developed for very specialized applications. Because of the large amount of computer resources which are required, this method has not been applied to general software development and maintenance applications.

HYPOTHESIS 2 - Analyzing maintenance activity data would assist with managing the maintenance process

Previous studies of the maintenance process (Ref 52,80-82 & 97) have proposed extensive maintenance activity data collection. The purpose of these proposals is the development of data base history of maintenance successes

and failures. Statistical analysis of this information could then be used by future managers to select the proper measures to complete maintenance tasks at lowest cost. Several questions were asked of software managers in the attempt to investigate the application of these proposals (Section IV, Questions 1-6). The questions attempted to identify additional management data which would assist managers with controlling the maintenance process. The large variance in responses showed that there was little agreement of what data should be collected or how the data should be used. Yet there was unanimous agreement that further maintenance management indicators would be helpful. Another point which was shared by the interviewed managers, was that the collection of activity data should not place a significant burden on the personnel performing maintenance tasks. Experienced personnel are already saturated with the control measures of configuration management and current tasks without an additional workload. A recommendation to adapt a maintenance activity collection and analysis scheme is made in Chapter V. This recommendation attempts to produce additional management information without burdening programmer/analyst personnel.

HYPOTHESIS 3 - Current management and programmer personnel have viable suggestions for improving the maintenance of existing systems

Several interview questions (Section III, Questons 4-7) were aimed at soliciting suggestions for improving the

software maintenance process. The answer that was given is that they are putting their ideas to work everyday. In fact, many of the advances that were discussed in hypothesis 1 of this section were developed and tested by in the field software maintenance personnel. Some obvious solutions for improving the software maintenance process are not possible within the realm of current DOD and Air Force policies. Changes to those policies and discussion of those solution methods will be present in Chapter V.

OBJECTIVE 3 Summary

The purpose of this thesis objective was to identify methods which are being developed or being used to improve the software maintenance process of Air Force weapon systems. Application of additional methods beyond those being used or developed were also investigated. The responses received coupled with the information obtained from the first two objectives provide the basis for the policy recommendations presented in the next chapter.

V. Summary and Recommendations

The purpose of this research effort was to improve the software maintenance of Air Force systems by studying current management and programming methods used by various systems. The success of this study is based on the perceptions and viewpoints obtained from the key personnel interviewed at each of the sampled systems. The findings outlined in Chapter IV present a descriptive picture of those systems which were investigated. The small sample size of this study limits the amount of anaylsis which can be extended to all weapon systems. Yet, the major findings of this study will act as an indicator of software maintenance conditions and practices in other Air Force systems. This chapter summarizes the major findings of this research, compares these findings to other systems, and presents guidelines for improving software maintenance. The accoplishment of the established thesis objective is summarized in light of these findings and recommendations.

Summary of Major Findings

The following research findings were formulated while evaluating several experimental hypotheses as the means of accomplishing the thesis objectives.

1. The basic elements of the software support environment differ between Air Force weapon systems.

2. Software maintenance programmers provide specialized software support functions.

3. System development efforts have not produced quality software products.

4. Software maintenance methods, standards, and goals are not adequate.

5. There is limited management of software maintenance costs.

6. There is a trade-off between efficient and effective software maintenance efforts.

These findings are based on the study of Air Force communications, avionics and ground radar sysems which have been operationally deployed for 1 to 20 years. These findings may not apply to computer software which is currently being developed or just recently delivered. These findings are an accumulation of the most significant conclusions formulated from the evaluation of the thesis hypotheses discussed in Chapter 4.

1. The basic elements of the software support environmental differ between Air Force weapon systems. The specialized missions of each Air Force weapon system, usually require one of a kind, real time hardware and software configurations. This causes the usage of a wide variety of assembly languages to attain the required efficiency with the different weapon system hardware components. The variety of hardware and programming language usage causes the utilization of various computer vendors software support tools. Different weapon system missions, different hardware, varied programming languages and different types of software maintenance tools all

contribute to making the software support environment different from one Air Force system to another.

To support the operational needs of the system. the maintenance of the software is closely managed by different system functional managers. A result of the differing maintenance management and workforce organizations is a varied approach to managing and performing software maintenance functions. Thus. when personnel are transferred into the software maintenance environment of a particular weapon system, the process of performing the general software support functions is consistent with those explained in conjunction with Figure 2-9; the exact management and programming elements which apply to completing these functions will differ between systems.

2. Software maintenance programmers provide specialized software support functions. Computer resources are required to fulfill similar weapon system functions; they act as the brain or controller of the system hardware. Because of this key role, a high degree of reliability and real time responsiveness is required of the software. As a result. software maintainers are subject to similar constraints. All software changes must be approved by several Air Force organizations and review boards. A majority (over 65 percent) of these changes are made using an assembly language. To insure reliability and performance requirements, the software code are tested separately and

with the integrated system. System tests are usually performed in a simulation environment and then during an operational test and evaluation. The process of approving, developing, and testing is accomplished for four major types of maintenance tasks:

1. Providing new software functions
2. Removing software defects
3. Tuning the software for greater efficiency
4. Providing greater system interface capabilities

Accomplishment of these diversified tasks takes a special combination of knowledge and experience. An effective software maintenance programmer/analyst must develop expertise in four critical areas: the weapon system, the software application, the software support facility and local standards and procedures for completing maintenance tasks. Obtaining personnel with this level of qualification is difficult undertaking. The Air Force either pays to develop inhouse experience through training programs or pays a contractor to provide this experience at about $65,000 per programmer/analyst.

3. <u>System development efforts have not produced quality software products.</u> Software maintenance personnel agree that software reliability is the most significant software quality measure that influences the amount of maintenance tasks. This study found that nearly 30 percent of the current maintenance workload is attributed to the

correction of errors that were created earlier in the software life cycle. The quality of the software documentation is also a critical issue which affects the level of maintenance workload. When evaluating the quality of the internal and external software documentation, and interviewed personnel from each system found fault in some portion of the documentation. The insufficiencies in software reliability and the level of documentation were largely attributed to the system acquisition philosophy. Evidently, the statement of work (SOW) which bound the contractors to developing software modules, did not adequately define deliverable software quality attributes. As a result, the Air Force purchases systems with delivered deficiencies. These deficiencies have to be corrected by the maintenance staff of lived with through the life cycle. The rising costs of performing maintenance tasks justifies the reconsideration of where these required software characteristics are implemented.

4. Software maintenance methods, standards, and goals are not adequate. The application of software development standards has varied from one system development project to another. Additionally, the software development practices vary between contractors. Even the software development styles of separate programmers within the same contractor organizations can widely vary. Plus. as new standard methods of developing program code are establish old methods

are used in decreasing levels. As a result, the conditions of the software products and documentation varied between the systems that were studied. Modern programming methods and conventions were used in varying degrees in these systems. It appeared that the trend of the observed systems was to extend the documentation and coding styles used during the original development. But when the original conventions were considered inadequate, some systems developed their own standards by which replacement code would be developed and implemented. These new standards seem to be based on modern programming concepts. as well as. the software support beliefs of lcoal experienced programmers and managers. These standards are stressed in training programs and some what enforced during program code testing and evaluation. Thus. different organizations and systems may be performing the same software support functions but the software is maintained at different levels of quality.

Most organizations have recognized the problem of inadequate maintenance programming methods, goals and standards. There are several major efforts underway to improve the current situation. Yet, the central question that once again occurs is: what level of software quality is required? Once this illusive issued can be resolved, a set of measurable software standards can be developed to control software development at all stages of the software life cycle.

5. There is limited management of software maintenance costs. The yearly expense of providing software maintenance functions was estimated to be in the range of $600 thousand to $1.5 million depending on the system. The distribution of where this money is applied was estimated by this study to be as follows:

50 percent - Software Support Facility
45 percent - Personnel Salaries
5 percent - Administrative and Travel Expenses

These software maintenance expenses were not readily available. They had to be extracted from contractor work-breakdown proposals and progress reports, partitioned from organizational budgets, and identified in purchase, lease, and support contracts for maintenance support computer resources. Personnel costs were estimated relative to current manning. Even though these cost estimates were able to be formulated, there appeared to be no attempt to collectively monitor or control these expenditures. Also, software development costs were not available. The reason given for the lack of development costs was that the software was only one component of an entire system acquisition and separate accounting of this component was not performed. Another reason was that the system was usually acquired by another organization (Air Force Systems Command).

The philosophy of recent DOD directives and Air Force regulations has been to emphasize that computer resources

(to include software) should be acquired and managed based on life cycle costs. Yet, it does not appear that Air Force management has collected the necessary cost data upon which to make these types of decisions.

6. <u>There is a trade-off between efficient and effective software maintenance efforts.</u> In order to effectively support the operational requirements of the weapon system, several general policy decisions influence how software maintenance will be managed and performed. For example, the software maintenance efforts of several of the studied systems are being managed directly by the functional using organization. This management approach attempts to get the personnel with the best system application knowledge to support software maintenance tasks. In some caes, the highest degree of the system knowledge and experience is not found in the Air Force. Thus contractor services support system and/or software maintenance functions. The decision as to who will manage or perform maintenance is based on which organization is best qualified to support the system requirements through the life cycle. This philosophy often causes software support organizations to be separately located and managed. This decentralization causes duplications of personnel and support equipment. Yet, the importance of mission requirements cause cost effective methods of performing maintenance to be considered second to effectively supporting the system. Maintenance facilities

of the future should strive to meet both goals: response effectiveness and process efficiency.

Recommendations

Many efforts are being developed and applied to increase management controls and programmer productivity within the maintenance process. Johnson (Ref 40) states that "productivity" is a function of selecting qualified people, establishing and adquate working environment, and providing proper management direction. With the explosion of new computer applications, the demand for experienced and qualified programmer personnel has exceeded the supply and the gap is predicted to get larger. For maintenance programming, qualified personnel must be developed on the job. The working environments needed are obtained by providing sufficient software tools and the procedures by which programmers are to apply them. Management directions takes the form of controlling the quantity and quality of software configuration changes, as well as, how resources will be committed to completing these changes. As development and maintenance workloads are increasing, current software technology advances are improving the rate of programmer productivity by three percent a year. Carlson (Ref) observes that actions must be to either reduce the maintenance workload or to increase productivity:

> "The cost of continuing to use armies of programmers to develop software is very high, especially during the maintenance phase of the life cycle.

> There are two main alternatives. One is to increase the productivity of software engineers so that a team of fixed size can handle more complicated requirements.
>
> The other is to stop treating software as a fixed product which is developed, maintained and discarded; and begin to develop evolutionary systems which adapt and become more sophisticated over long periods of time."

On the basis of the research completed in this thesis, several recommendations have been developed with the intent of increasing maintenance productivity and reducing the current workload. These recommendations specifically apply to Air Force weapon systems of the general type surveyed. These suggestions will not cure current maintenance difficulties instantly. However, they will provide a further means of correcting problems that will continue to persist until some action is taken. The recommendations outline methods for the:

1. Development of experienced software maintenance personnel,
2. Management of system life cycle growth,
3. Development of maintenance standards,
4. Management control of maintenance resources, and
5. Further study.

Recommendations for accomplishing these goals are discussed below.

1. Development of experienced software maintenance personnel. The critical issued confronting maintenance

managers is how to develop and retain experienced personnel. The Air Force has not adequately answered this questions. Usage of a system escort team might be a viable method of developing the needed expertise. Ordinarily the Air Force develops the system specifications and the contractor builds the system. The level of knowledge of the system transfers from the Air Force to the contractor and is never reestablished. Contractors are paid to develop and maintain systems and at the same time gain the greatest level of knowledge and experience with the system. This leaves the Air Force in a vulnerable position both at contract negotiation and in adequately understanding the capabilities of the software. The system escort team philosophy calls for Air Force participation with software development as the system is being developed by the contractor. This could be done by assigning a group of Air Force personnel to the contractor as "free" resources to be used with software development. This basic concept is already being used with contractors in other functional areas. This concept could be extended to providing personnel to software maintenance contractors. The result would be a group of Air Force personnel which have gained a high degree of experience with the system in the software arena. The next concern is how to retain this expertise. Personnel which develop high technical knowledge of particular software system will be most productive on that system because of the vast difference between weapon system hardware and software

elements. Two suggestions are offered for retaining experienced personnel: longer period of assignment and system-technical career paths. Longer assignments would allow Air Force personnel to spend more time on the job after developing experience. This would increase the average productivity and decrease the commitment to large and extensive training programs. Currently, once personnel attain technical competence and productivity, they are promoted to management. Contractors such as TRW, General Dynamics, Raytheon, and Boeing have long realized that retention of technical experience over time is a critical concern. One method used by these companies is the formulation of a technical career path. This career plan allows personnel to progress in salary and grade and still perform technical functions. Even though some turn over is essential to provide an influx of new ideas and skills, the Air Force's technical software and systems experience could be retained over the life of the system.

2. Management of system life cycle growth. Currently, systems are developed to meet Air Force specifications. The software is one functional component of the system. The software and corresponding hardware is developed to meet the minimum functional specification. As a result, when systems are deployed and enhancements are requested, there are few available system resources to handle the increased requirement. This study showed that 40 percent of the

software maintenance tasks are enhancements. Implementation of these enhancements is often limited by memory of speed capabilities of the hardware. Knowing that software growth is a fact of life, the Air Force should begin to plan for this growth. One way this can be accomplished is to change the philosophy of system and software acquisitions. The Air Force should develop specifications which require contractors to develop systems which can be expanded to meet projected growth requirements. Then once this capability is established, proper control of growth will have to be provided. The major instrument for controlling the level of present enhancements is the limitations of available resources. New functional enhancements can not be implemented until an old functional capability is removed. Resource limitations force priorities to be established. Once the resources are made available through acquisition planning, other mechanisms for limiting growth will have to be found. Stricter configurations management is the obvious solution; making it work is another.

3. Development of maintenance standards. The diversity of current approaches to developing and maintaining software affects the quality of code and the maintainer's ability to work with it. Current DOD and Air Force standards are meant to be all-inclusive. These standards do not reflect current advances in software engineering practices. As a result the standards are

applied at the judgement of managers. An effort should be made to redevelop standards and guidelines which outline minimal not maximal requirements. These standards should reflect measureable software qualities which easily enable maintenance tasks to be completed.

Much hope has been placed in the development of the DOD common programming language ADA. It is believed that a common programming language will provide a mechanism for imposing some constraints to ensure proper language usage. Indeed, the usage of ADA could result in the standardization of the entire software maintenance facility, to include standards and procedures. This type of commonality will provide the structure for defining a common approach to developing software and insuring quality. The standards developed for current systems cannot be ignored. It is unrealistic to think that a system which currently has one million lines of code will be converted to another language. So standards should be developed to apply to these systems also. These standards would control the five percent replacement code which is developed each year.

4. Management control of maintenance resources. When systems are conceived there is contention for control of the software maintenance function between the user command and Air Force Logistics Command (AFLC). Both organizations have an interest in performing the software support function. AFLC has greater experience with providing maintenance

functions for all types of Air Force material items to include computer software. With this experience AFLC believes that they can provide the more efficient software support. The user command believes that more responsive and therefore more effective control can be established if management of the software support effort remains in house. Mission requirements often dictate that software maintenance be performed by the user command. As a result, many separate organizations have control of software support functions. One suggestion to reduce the impact of diverse management control is to establish a central organization which is responsible for establishing the software maintenance environment. This organization would attempt to share management and technical expertise between separate systems. This organization would provide a focal point for reducing duplicate expenditures of support resources.

At the local level management should provide a control over maintenance expenditures. Managers should selectively allocate resources to assist with software support based on past proven performance. Currently, managers have only their limited personal experience from which to draw from. As suggestion for increasing the amount management data available for making decisions is to develop a maintenance activity data base. By continuously measuring the presence, magnitude, and performance of several factors in relation to the required level of effort, a history of maintenance productivity could be established. A statistical analysis

of this information could then be used by managers to select the proper measures to complete maintenance tasks at lowest cost. A more basic recommendation for controlling software maintenance expenses is an adequate system of cost accounting and an active management commitment for monitoring these costs.

5. Further study. Software maintenance is a rapidly expanding field of study. Many authors believe that in time software engineering concerns will widely emphasize the development of advanced practices for accomplishing maintenance tasks. As of yet, relatively little study has been accomplished in this area. Two suggestions for further study of software maintenance as it relates to Air Force weapon systems are summarized below:

Study 1: System/Software Specification. This study would take a current system which is the conceptual phase and develop an alternate system specification document parallel to normal specification development. This specification would bind the contractor to deliverable software quality measures and provide system resources for planned software growth. It also would be interesting to evaluate the cost differences between this approach and current methods.

Study 2: Maintenance Activity Data Base. This study would design and implement a method of collecting maintenance activity information and resource costs. A

management approach to utilizing this data would also be developed.

Comparison of Findings

Analysis of the interview data shows that there are many similarities and differences between the software maintenance of Air Force systems and industrial systems. Air Force software was fundamentally different characteristics. The average life span of weapon system software exceeds ten years. Program code normally consists primarily of assembly languages. Each system has unique hardware and software configurations. The software is only one component of a complete system. Air Force systems require a much higher degree of reliability, thus software testing is a much more significant activity. This concern for reliable software has caused the Air Force to take a formalized and progressive approach to completing software support. In fact, the Air Force has been performing many software support functions which are still regarded as improvement recommendations for industrial systems. Techniques such as unitary maintenance organizations, formalized training of personnel, software quality assurance, configuration management programs, scheduled implementation of software changes, and usage of software life cycle prediction models are common place within Air Force software support efforts.

The problem areas identified with this study are very simular to those being experienced by industrial systems. The emergence of software maintenance as a significant management and programming effort has left all organizations lacking adequate personnel experience, and maintenance standards and procedures. Organizations have also found that the current state of documentation and code quality is at a level far below what is required. This research suggests methods of handling these problem areas.

Thesis Accomplishments Summarized

The purpose of the first thesis objective was to identify factors which influence the level of maintenance costs. This was accomplished by linking the level of maintenance workload with the amount of resources required to complete the workload. The commitment of resources is influenced by four major factors: experience of personnel, system age, documentation and program code quality, and the level of user enhancements. All other factors were seen as either constants or as factors not significantly affecting the maintenance workload or the ability to complete it.

The second thesis objective was to identify the Air Force policies which cause excess resource commitments to software maintenance. The Air Force current position on four major issues were discovered to influence excess resource expenditures: turnover of personnel, system/software development methods, software development

and maintenance standards, and resource management. The Air Force has made significant progress with controlling software maintenance activities. The Air Force's policies surrounding these four issues cause excess amount of personnel training, duplication of personnel and computer resources, and time spent completing maintenance tasks. The policies also cause software to be delivered in a state which is not condusive for software growth and necessary maintenance changes.

The third thesis objective was to provide constructive recommendations for improving the software maintenance process of Air Force weapon systems. The recommendations attempted to reduce the effect of the problem areas identified in the first two thesis objectives. These recommendations were directed at:

1. developing and retaining software maintenance experience with the Air Force,
2. planning and managing for system/software life cycle growth,
3. developing software development standards which better link software development with software support,
4. increasing control over maintenance resource commitments, and
5. furthering study of the software support process of the Air Force systems.

Thesis Conclusion

In the past, improvement suggestions could easily be disregarded. Weapon systems and their associated software did not undergo the same degree of change. Indeed, the dependence on software to complete vital weapon system functions was less than today. Future commitments to software are predicted for substantial increases. Increased commitments will be directed at supporting the growing number of computer software - dependent Air Force systems and supporting the ever lengthening life span of deployed systems. Commitments will be in the form of personnel and computer resources needed keep the software responsive to the changing operational system requirements. A continuation of software support under current Air Force policies will cause excess consumption of these resources. Left unattended, this will result in either a massive resource expenditure or a limited maintenance response capability. Actions need to be taken in order to avoid these extreme situations. The recommendations offered in this thesis provide specific guidance as to where improvement should be directed.

With the growth of software maintenance as a significant concern, we are witnessing the emergence of a new subscience within software engineering: software maintenance engineering. Currently, this field consists of the development of management and programming practices which affect the support of software after its initial

development. This thesis work has profiled how these practices apply the the support of Air Force weapon system software. This work has rediscovered that technology often progresses faster than management and programmers can react. The increased utilization of computer software has created a growing software maintenance workload. Managers are searching for the best approaches to effectively and efficiently meeting this workload. The findings from the study of thirteen Air Force systems has contributed to the understanding of current approaches and has helped to identify specific goals for future improvements.

Bibliography

1. Aeronautical Systems Division. Software Acquistion Engineering Guide Books. Wright-Patterson AFB OH.

   Regulations, Specifications and Standards - ASD-TR-78-6
   Reviews and Audits - ASD-TR-78-7
   Software Quality Assurance - ASD-TR-78-8
   Configuration Management - ASD-TR-79-5024
   Computer Program Documentation Requirements - ASD-TR-79-5025
   Statements of Work and Requests for Proposal - ASD-TR-79-5026
   Requirements Analysis and Specificaiton - ASD-TR-79-5027
   Verification, Validation, and Certification - ASD-TR-79-5028
   Microprocessors and Firmware - ASD-TR-80-5021
   Software Development Planning and Control - ASD-TR-80-5022
   Software Testing and Evaluation - ASD-TR-80-5023
   Contracting for Software Acquistion - ASD-TR-80-5024
   Software Cost Analysis and Estimating - ASD-TR-80-5025
   Supportable Airborne Software - ASD-TR-80-5026
   Software Development and Support Facilities - ASD-TR-800-5027
   SAE Guidebooks - Application and Use - ASD-TR-80-5028

2. Air Force Test and Evaluation Center. Software OT&E Guidelines Vol I-IV, Kirtland AFB NM. 1981

3. Alford, M. "A Requirements Engineering Methodology for Real Time Processing Requirements" in IEEE Trans. Software Engineering. 60-68. Jan 1977

4. Bachman, C. and Bouvard J. "Architecture Definition Technique: Its Objectives, Theory, Process, and Facilities and Practice" in Proc. ACM SIGFIDEF Data Description Access and Control. 257-305. 1972

5. Balzer, R. Automatic Programming. Technical Memo, Information Science Institute, USC, 1972.

6. Belady, L.A. and Lehman, M.M., "The Characteristics of Large Systems," Research Directions in Software Technology, edited by P. Wegner, Cambridge, MA, MIT Press, 1979.

7. Bert, D. "Problem Specification and Algorithmic Programming" at International Computing Symposium, 1977. 1977

8. Black, R.K.E., Katz, R., Gray, M.D., and Curnow, R.P. "BCS Software Production Data Final Technical Report", RADC-TR-77-116. March 1977

9. Boehm, Barry M., "Software Engineering: R&D Trends and Defense Needs," Research Directions in Software Technology, edited by P. Wegner, Cambridge, MA, MIT Press, 1979.

10. Boehm et al., Characteristics of Software Quality. New York: North Holland, 1977.

11. Boehm, B., et at. "Quantitative Evaluation of Software Quality," Proceedings, 2nd International Conference of Software Engineering, October 1976.

12. Canning, R.G., "That Maintenance Iceburg" in EDP Analyzer. 1972

13. Dickover, M., McGowan, C., and Ross, D.T. "Software Design Using SADT" in Proc. Nat. ACM Conf., 125-137, 1977.

14. DeRose, B.C. and Nyman, T.H., "The Software Life Cycle - A Management and Technological Challenge in the Department of Defense", IEEE Transactions on Software Engineering, 4:309-320 (July 1978).

15. De Roze, "Defense Systems Software Management Plan", Office of the Assistant Secretary of Defense (Installations and Logistics), DDC Accession Number AD-A022 558, March 19, 1976.

16. Department of the Air Force. AFR 800-14, Volume I, "Management of Computer Resources in Systems." Washington, D.C.: September 1975.

17. _____. AFR 800-14, Volume II, "Acquisition and Support Procedures for Computer Resources in Systems." Washington, D.C.: September 1975.

18. Department of Defense. Directive 5000.1, "Major System Acquisition." Washington, D.C.: January 1977.

19. _____. Directive 5000.29, "Management of Computer Resources in Major Defense Systems." Washington, D.C. April 1976.

20. _____. Instruction 5000.1, "Acquisition of Major Defense Systems." Washington, D.C.

21. _____. Directive 500.1. Acqusition of Major Defense Systems. Washington: Government Printing Office, December 1975.

22. _____. Directive 5100.40. Responsibility for the Administration of the DOD Automatic Data Processing Program. ASD(C). Washington: Government Printing Office, August 1975.

23. _____. Directive 4155.1. Quality Assurance. Washington: Government Printing Office, February 1972.

24. _____. Directive 5010.19. Configuration Management. Washington: Government Printing Office, April 1970.

25. _____. Instruction 5010.21. Configuration Management Implementation Guidance. Washington: Government Printing Office, January 1969.

26. _____. Standard 490. Specification Practices. Washington: Government Printing Office, October 1968.

27. _____. Standard 480A. Configuration Control. Washington: Government Printing Office, April 1978.

28. _____. Standard 483(USAF). Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs. Washington: Government Printing Office, December 1970.

29. Donahoo, John D. and Swearington, Dorothy R., "A Review of Software Maintenance Technology," RADC-TR-80-13, 1980, AD-A082 985/3.

30. Fink, R.C. "Major Issues Involving the Development of an Effective Management Control System for Software Maintenance," Proceedings COMPSAC 1977, Chicago, IL. 533-538. November 1977.

31. Gilb, Software Metrics. Cambridge, MA: Winthrop, 1976.

32. Goldberg, J. "Proceedings of a Symposium on the high cost of software," held at the Naval Postgraduate School, Monterey, CA. DDC Accession Number AD-770 121, Sept 17-19, 1973.

33. Goldberg, P. "Automatic Programming" in Lecture Notes in Computer Science. New York: Springer Verlag, 1975.

34. Gustafson, G.G., and Kerr, Roberta J., "Some Practical Experience with Software Quality Assurance Program," Com of ACM, 25:1.4-12, Jan 1982, pp 4-12.

35. Halstead, M.H., Elements of Software Science. Elsevier North Holland, New York, 1977.

36. Hamilton, M. and Zeldin, S., "Higher Order Software A Methodology for Defining Software" in IEEE Trans. Software Engineering. 9-32, June 1976.

37. Hammer, M., Howe, W.G., Kruskal, V.J., Wladawsky, I., "A Very High Level Programming Language for Data Processing Applications" in Communications of the ACM. 20:11:832-841. November 1977.

38. Hantler, S.L. and King, J.C., "Introduction to Program Correctness" in Journal of ACM. 331-353. Sep 1976.

39. Huff, S. and Modnick, S., "An Extended Model for a Systematic Approach to the Design of Complex Systems" NTIS Report ADA 058565. 1978

40. Johnson, D., Kolberg, C., and Sinnamon, J., "A Programmable System for Software Configuration Management," Proceedings COMPSAC 1978, Chicago, IL, November 1978, pp. 402-407.

41. King, James, "Program Correctness: On Inductive Assertion Methods" in IEEE Transactions of Software Engineering. 6:5: 465-479. Sep 1980.

42. Kossiakoff et al., "DOD Weapon Systems Software Management Study," Applied Phystics Laboratory, The Johns Hopkins Univ., Laurel, MD, Rep. SR-75-3, June 1975.

43. Knight, B.M., "Software Quality And Productivity," Defense Systems Management Review, 1(7-8): 54-65 (Autumn 1978).

44. Lehman, Meir M., "Programs, Life Cycles, and Laws of Software Evolution," Proceedings of IEEE, 9: 1061-1076, (September 1980).

45. Lehman, M.M. and Parr, F.N., "Program Evoluation and and Its Impact on Software Engineering", Proceedings of the 2nd International Conference on Software Engineering. San Francisco, 350-357, October 1976.

46. Lientz, B.P. and Swanson, E. Burton. "Problems in Application Software Maintenance," Communications of the ACM, 11: 763-769, (November 1981).

47. Lientz, B.P., E.B., Swanson, and G.E. Tompkins. "Characteristics of Application Software Maintenance," Communications of the ACM 21, 466-471, 1978.

48. Lientz, B.P. and E.B. Swanson. "The Impact of Development Productivity Aids on Application Software Maintenance," Proceedings, Conference on Application Development Systems, San Jose, CA, March 9-11, 1980.

49. Lientz, B.P. and E.B. Swanson. "Software Maintenance: A User/Management Tug-of-War," Data Management, 26-30, 17 April 1979.

50. Lientz, B.P. and E.B. Swanson. "Problem Factors and Determinants in Application Software Maintenance," Information Systems Working Paper 7-79, Graduate School of Management, University of California, Los Angeles, March 1979.

51. Lientz, B.P. and E.B. Swanson. "Discovering Issues in Application Software Maintenance," Data Management, 16 September 1978.

52. Lientz, B.P. and E.B. Swanson, Software Maintenance Management, Addison-Wesley Publishing Company, Phillipines, 1980.

53. Lindhorst, W.M., "Scheduled Maintenance of Applications Software" in DATAMATION. 87-89. May 1973.

54. Littlewood, B. (1975), "A Reliability Model for Markov Structured Software," Applied Statist., Vol. 24, pp. 172-177.

55. Littlewood, B. and Verrall, J.L. (1973), "A Bayesian Reliability Growth Model for Computer Software," Applied Statist., Vol. 22, pp. 332-346.

56. Liu, Chester C., "A Look at Software Maintenance," Datamation, 51-55, Sep 1976.

57. Manley, J., "Findings and Recommendations of the Joint Logistics Commanders Software Reliability Work Group (SRWG Report)." Headquarters, Air Force Systems Command (XRF), Andrews AFB, MD, Final Rep., Vol. I and II, Nov 1, 1975.

58. Manna, Z., Ness, S., and Vuillemin. "Inductive Methods for Proving Properties of Programs" in Proc. ACM Conference on Proving Assertions about Programs, SLGPLAN Notices. 7:27-50. Jan 1972.

59. McCabe, T, "A Complexity Measure," Proceedings of the 2nd International Conference on Software Engineering, IEEE, 1976.

60. McCabe, T. J., and Frederick Stern., "Use of Metrics to Measure Quality," Conference Proceedings from the DPMA National Symposium on Effective Methods of EDP Quality Assurance, Chicago, IL, April 1-3, 1981.

61. McCall, J.A., "An Introduction to Software Quality Metrics" in Softwrae Quality Management, edited by Fisher and Cooper. New York: Petrocelli Books, Inc., 1979.

62. McCall, J. and Walters, G., "The Development of Metrics for Software R&M," 1978 Proceedings of the Annual Reliability and Maintainability Symposium January 1978.

63. McCall, J., Richards, P., Walters, G., "Metrices for Software Quality Evaluation and Prediction," Proceedings of Second Summer Software Engineering Workshops, NASA/Goddard Space Flight Center, Sep 1977.

64. Mercer, B., "Weapon System Software Acquisition and Support: A Theory of System Structure and Behavior" Unpublished Master's Thesis. Air Force Institute of Technology, Wright-Patterson AFB OH, March 1982.

65. Miller, E., and Howden, W.E., Tutorial: Software Testing and Validation Techniques. IEEE Comptr. Soc., Los Alamitos, CA, 1978.

66. Mills, H.D., "Software Development," Research Directions in Software Technology, edited by P. Wegner, Cambridge, MA, MIT Press, 1979.

67. MITRE Corp., "DOD Weapon System Software Acquisition and Management Study," the MITRE Corp., Bedford, MA, Rep. MTR-6908, Vol. I and II, June 1975.

68. Murch, W.G., Operational Test and Evaluation of Software, National Aerospace Electronics Conference Proceedings, Dayton, Ohio, pg 1390, 1981.

69. Munson, J.B., "Software Maintainability: A Practical Concern for Life-Cycle Costs," Computer, 103-109 (November 1981).

70. Myers, G.T., The Art of Software Testing. Wiley New York, 1979.

71. Myers, G.J., Reliable Software Through Composite Design, Petrocelli/Charter, 1975.

72. Myers, G.J., Software Reliability: Principles and Practices, John Wiley & Sons, New York, 1976.

73. Myers, G.J., "Characteristics of Composite Design," DATAMATION, September 1973.

74. Myers, W., "The Need for Software Engineering," Computer, February 1973.

75. Myers, W., "COMSAC Wrap Up" in Computer. 62-70, January 1979.

76. Musa, J.E. (1975), "A Theory of Software Reliability and its Application," IEEE Trans. on Software Engineering, Vol. SE-1, No. 3, pp. 312-327.

77. Nyman et al, "Defense System Software R&D Technology Plan," R&D Technology Panel to the Management Steering Committee for Embedded Computer Resources, Office of the Under Secretary of Defense for Research and Engineering, Department of Defense, Rep., November 1977.

78. Okumoto, K., and Goel, A.L., (1978), "Availability Analysis of Software Systems Under Imperfect Maintenance," RADC-TR-78-155, Vol, III.

79. Okumoto, K., and Goel, A.L., (1978), "Classical and Bayesian Inference for the Software Imperfect Debugging Model," RADC-TR-78-155, Vol. II.

80. Orr, K., Structure Systems Development, New York, Yourdon Inc., 1977.

81. Peercy and T. Paschich, "Software Maintainability Analysis Program User's Manual," BDM/TAC-78-697-TR, December 1978.

82. Peercy, "Software Maintainability Evaluation Guidelines Handbook," BDM/TAC-78-687-TR, December 1978.

83. Peercy, D.E., "A Software Maintenance Evaluation Methodology," IEEE Trans. Software Engineering, Vol. SE-7: 343-352, July 1981.

84. Prywes, N.S., Pnueli, A., Shastry, S., "Use of a Nonprocedural Specification Language and Associated Program Generator in Software Development" in ACM Transactions on Programming Languages and Systems, 1:2:196-217, October 1979.

85. Prywes, N.S., "Automatic Generation of Computer Programs" in Advances in Computers. Academic Press: New York. 1977.

86. Punter, M., "Programming for Maintenance," Data Processing, September-October 1975, 292-294.

87. Putnam, L.H., "General Empirical Solution to the Macro Software Sizing and Estimating Problem," IEEE Transactions on Software Engineering, July 1978.

88. Putnam, L.H., and Fitzsimmons, A., "Quantative Management: Software Cost Estimating" presented at IEEE Computer Society COMSAC 77, Chicago, IL., 129-137. November 1977.

89. Richards, P.K., G. Walters, and J.A. McCall., "Factors in Software Quality," Three volumes, NTIS, AD-A049-014, AD-A049,015, AD-A049-055., November 1977.

90. Rin, N.A., "Automatic Generation of Data Conversion Programs Using a Data Description Language," PHD Dissertation in Computer Science, Vnn of Penn, Philadelphia, PA, 1976.

91. Schneider, J., "A Preliminary Calibration of the RCA PRICE-S Software Cost Estimation Model." Unpublished master's thesis. Air Force Institute of Technology, Wright-Patterson AFB OH. July 1976. AD A046808.

92. Schick, G.J., and Wolverton, R.W. (1972), "Assessment of Software Reliability," McDonnell-Douglas Astronautics Company Paper WD 1872.

93. Sharpley, W.K., "Software Maintenance Planning for Embedded Computer Systems," COMPSAC 77, 520-527, November 1977.

94. Structured Programming Series, USAF Rome Air Development Center, Vols. 1-15, July 1975.
DDC Accession Numbers Follows:

"Programming Language Standards," AD-A016-771.
"Pre-Compiler Specifications," AD-A018-046.
"Pre-Compiler Program Documentation," AD-A013-255.
"Data Structuring," AD-A015-794.
"Program Support Library Requirements," AD-A003-339.
"Program Support Library Program Specifications," AD-A007-796.
"Documentation Standards," AD-A016-414.
"Program Design Study," AD-A016-415.
"Management Data Collection and Reporting," AD-A008-640.
"Chief Programmer Team Operations," AD-A008-861.
"Estimating Software Resource Requirements," AD-A016-416.

"Training Materials," AD-A026-947.
"Software Tool Impact," AD-A015-795.
"Validation and Verification," AD-A016-668.
"Final Report," AD-A020-858.

95. Swanson, E.B., "The Dimensions of Maintenance," Proceedings, 2nd International Conference on Software Engineering, San Francisco, 13-15 October 1976, 492-497.

96. Swanson, E.B., "On the User-Requisite Variety of Computer Application Software," IEEE Transactions on Reliability, R-28, August 1979, 221-226.

97. Swanson, E.B., "The Dimensions of Maintenance," Proceedings 2nd International Conference on Software Engineering, 492-497, October 1976.

98. Swinson, G.E., and Jones, S.D., "Standard Software Support Facility Evaluation - Final Report," BDM/TAC 80-693-TR, BDM Corporation Report, Nov 1980.

99. Trainer, W.L., "Software From Satan to Savior", Proceedings NAECON Conference, May 1973.

100. Trivedi, A.K., and Shooman, M.L. (1975), "A Many-State Markov Model for the Estimation and Prediction of Computer Softwre Performance Parameters," Proceedings: 1975 International Conference on Reliable Software, pp. 208-220.

101. Walston, C.E., and Felix, C.P., "A Method of Programming, Measurement and Estimation" in IBM Systems Journal. 16:1:54-73. 1977.

102. Warnier, J., Logical Construction of Programs, Leiden Germany: Stenfert Kroese, 1974.

103. Watkins, M.L., "A Technique for Testing Command and Control Software" in Communications of ACM. 25:4: 228-232. April 1982.

104. Winograd, Terry., "Beyond Programming Languages" in Communications of the ACM, 22:7:391-398, July 1979.

105. Wolverton, R.W., "The Cost of Developing Large-Scale Software," IEEE Trans. on Computers, June 1974.

106. Wong, K., and Engelland, J., "Operational Software Concept: A New Approach to Avionics Software" in Proc. AIAA Digital Avionics Systems Conf., 1975.

107. Yau, S.S., and Collofello, J.S., PERFORMANCE RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE, RADC-TR-80-55, by Northwestern University, Dept, of Electrical Engineering and Computer Science for RADC (ISIS), Griffiss AFB NY, 1980, (NTIS No. AD-A084-351).

108. Yau, S.S., and Collofellow, J.S., "Some Stability Measures for Softwre Maintenance," Proceedings of COMPSAC - IEEE Computer Society 3rd International Computer Software and Appl. Conf., Chicago, IL, 6-8 November 1979.

109. Yourdon, E., Techniques of Program Structure and Design, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

APPENDIX 1

STRUCTURED INTERVIEW OF AIR FORCE WEAPON SYSTEM SOFTWARE MAINTENANCE

PURPOSE

The purpose of this study effort is to gather information about the maintenance of Air Force weapon system software. This information will be used to:

1. Identify management and programming methods which are used during the maintenance life cycle.

2. Identify perceived problem/solutions within the maintenance process.

3. Assess the factors which influence software maintenance costs.

4. Develop and evaluate a system for gathering maintenance cost/performance data.

ORGANIZATION

The questions which will be asked during the structured interview are partitioned into four sections:

Section I: General System Description

Section II: Maintenance Effort - Facts

Section III: Maintenance Effort - Perceptions

Section IV: Maintenance Data Evaluation

SECTION I: GENERAL SYSTEM DESCRIPTION

1. Name of person interviewed:

2. Role in maintenance process:

3. Office symbol/address:

4. Name of Air Force weapon system:

5. Basic function of system:

6. Software development information -

   a. Cost of entire system:

   b. Cost of software:

   c. Cost estimation method:

   d. Development time period:

   e. Expected software lifespan:

   f. Who completed software development:

   g. Which methods and tools were used in development:

Methods	Tools
Structured Programming	Automated Flowcharting
Decision Tables	Automated Documentation
Chief Programmer Team	Data Base Dictionary
Structured Walkthrough	Program Design Language
Others (Specify):	Others (Specify):

h. What programming standards were used during software development:

i. Describe the hardware that was acquired for system operations:

j. Describe the hardware that was acquired for system support and maintenance:

7. Software Product Information -

a. Number of program modules originally delivered:

b. Number of lines of code originally delivered:

c. Programming language used as percent of total -

assembler (identify assembler name)
FORTRAN
JOVIAL
OTHER-(Specify)

Total 100%

SECTION II: MAINTENANCE EFFORT - FACTS

1. Specify which functions are in your organization's definition of software maintenance and software development.

   (Specify a M for maintenance and a D for development)

   a. Removing defects in which the program logic was faulty with the result that the program did something other than what the system and subsystem specification intended.

   b. Tuning the software to make it more efficient (less machine time. less core).

   c. Providing new functions requested by the user.

   d. Making modifications as a result of the requirement to interface with other systems or other environments.

   e. Others? (Specify)

2. Of the functions (above) that are attributed to maintenance, breakdown the percentage each task is to total software maintenance.

a.	%
b.	%
c.	%
d.	%
e.	%
Total	100%

3. Who has been responsible for completing software maintenance functions:

4. Describe the growth in software by number of lines of code since deployment.

Number of Lines

Time in years

5. Describe the cost of software maintenance support.

Costs

Time in years

6. What is the total number of individuals which are currently assigned (in whole or part) to software maintenance:

Personnel Type	Number	Percentage of Time
Clerical		
Analyst/Programmers		
Managers/Staff		
Other:		

7. Describe what percentage of time a manager spends on the following functions:

    a. Readiness management (planning/estimating future maintenance support requirements)

    b. Configuration management (change control)

    c. Software Quality assurance

    d. Scheduled Maintenance Management

    e. Other:

    100% Total

8. Describe what percentage of time a programmer spends on the following functions:

    a. Problem analysis and solution design

    b. Coding

    c. Configuration Control Process Interface

    d. Documentation changes

    e. Testing and implementation

    f. Other functions

    100% Total

9. Of the current systems analysts and programmers:

    a. How many worked on the initial development of the software:

    b. What is the average number of years of maintenance experience:

    c. What type of training have these personnel received:

Before Assignment to project:

After Assignment to project:

10. a. Describe the automated tools which are utilized for maintenance development:

b. How/why were these selected:

c. What is the cost of development tools:

11. a. Describe the automated tools which are utilzed for maintenance development testing:

b. How/why were these selected:

c. What is the cost of testing tools:

12. a. Describe other computer resources which are dedicated to software maintenance:

b. How/why were these selected:

c. What is the approximate cost:

13. What further resources are needed?

14. On what basis are the following system resources committed/allocated to support maintenance functions: programmers, clerical personnel, computer time, office space/supplies, and automated development and test tools?

15. Indicate which methods are utilized and describe extent of usage.
Structured programming -

Structured walkthroughs -

System Librarian -

Design specification (HIPO,PDL,SADT) -

Test Data Generators -

Decision Tables -

Flow Charting -

Data Base Dictionary -

16. Describe the standards and procedures of each stage of the maintenance process below:

Design specification -

Describe format used

Describe Approval procedure

CODING and Testing -

Describe procedures used

Format of documentation

Who reviews the coding/documentation

Describe the testing procedure

Describe the approach to integration

Who reviews the integration

17. Before a software change is implemented, how many different offices/boards must coordinate on the action?

SECTION III: MAINTENANCE EFFORT - PERCEPTIONS

1. Assign a quality score for the current status of each of the following entities. If none of the available descriptions accurately describes the entity then give your own description.

a. DOCUMENTATION OF THE SOFTWARE score:

0 = none
1 = some in line
2 = in line with flowcharts
3 = in line, flowcharts, manuals, design documents but each are incomplete, poorly organized/written, and noncurrent
4 = in line, flowcharts, manuals, design documents but not complete
5 = current/effective/complete documentation
6 = other:

b. PERSONNEL CAPABILITIES score:

0 = insufficient number and quality personnel
1 = insufficient orientation or training to application
2 = insufficient maintenance programming experience
3 = insufficient number of committed personnel
4 = qualified, capable programmer workforce
5 = other:

c. MAINTENANCE BUDGET score:

0 = poorly insufficient - not adequately planned
1 = little less than needed
2 = about right but poorly applied
3 = about right but adequately applied
4 = too well padded
5 = other:

d. MANAGEMENT MAINTENANCE CONTROLS score:

0 = poor controls - little positive effect
1 = effect of current controls not known
2 = current controls are effective, but further controls are needed
3 = current controls are adequate
4 = current controls are too restrictive
5 = other:

e. SOFTWARE QUALITY score:

Range of Score Values 0-5

Low Description: Code is not structured, exhibits poor flow control (not top down). Code contains several Khiges. Errors occur fairly often. Code can only be maintained by those who have been working within for some time.

Medium Description: The complexity of the programming language, program logic, or structure of code limits those who can maintain it.

High Description: Program modules are coded using standard AF language and standard language features. Program design structure is modular. Structured program constructs are used throughout. Code demonstrates high level of reliability.

2. In your judgement, to what extent have the following been (or are) problems in maintaining the application system you have described?

	None at All				Major Problem
	5	4	3	2	1
a. Turnover of maintenance personnel					
b. Quality of application system documentation					
c. Changes made to system hardware and software					
d. User demand for enhancements and extensions to application systems					
e. Skills of maintenance programming personnel					
f. Quality of original programming of application system					
g. Number of maintenance programming personnel available					
h. Competing demands for maintenance programming personnel time					
i. Lack of user interest in application system					
j. Application system run failures					
k. Lack of user understanding or application system					

None at All

Somewhat Minor

Minor Problem

Somewhat Major

Major Problem

5 4 3 2 1

l. Storage requirements of application system programs

m. Processing time requirements of application system programs

n. Motivation of maintenance programming personnel

o. Forecasting of maintenance programming personnel requirements

p. Maintenance programming productivity

q. System hardware and software reliability

r. Data integrity in application system

s. Unrealistic user expectations

t. Adherence to programming standards in maintenance

u. Management support of application system

v. Adequacy of application system design specifications

w. Budgetary pressures

x. Meeting scheduled committments

OTHERS: (please Specify)

y.

z.

3. Rank the top three and bottom three entities as they affect the level of maintenance support needed (measured in resources committed to maintenance).

Entity

a. Budget available
b. Size of software (lines of code)
c. Code and documentation quality
d. Experience of maintenance Programmers
e. Level of user enhancements
f. History of error occurence
g. System Age
h. Management controls to be used
i. Maintenance tools to be used

4. You have identified these as problem areas:

a.

b.

c.

d.

In what ways could these problems be reduced:

a.

b.

c.

d.

5. You have identified these areas which managers spend a majority of their time:

   a.

   b.

   c.

   What could be done to increase the efficiency of these operations?

   a.

   b.

   c.

6. You have identified these areas which programmers spend a majority of their time:

   a.

   b.

   What could be done to increase the efficiency of these operations?

   a.

   b.

7. What could be done to reduce the current resouce commitment (costs) to software maintenance?

SECTION IV: MAINTENANCE DATA EVALUATION

1. Are all user requests for changes to the application system logged and documented according to an established procedures?
   If so. what is the procedure

   How many requests were recorded during the past twelve months?
   Of these requests. how many have been satisfied by changes to the application software
2. Are all troubles encountered with the operational processing of the application system logged and documented?
   If so. what is the procedure

   How many trouble reports were recorded in the last twelve months?
3. Are there accounting procedures for accumulating cost data for each of the following entities as they relate to software maintenance?

Entity	Y/N
Personnel	
Clerical	
Management	
Programmers	
Software Tools	
Machine Time	
Other resources	

Explain procedures/methods:

4. For those activities for which cost data is collected, give the cost estimations for the last 12 months.

Entity	Cost Estimation

5. How are the logs and cost data used?

6. Which of the following values would be helpful in planning and managing the maintenance effort:

   a. Determination of what types of maintenance costs most at yours and other installations.

   b. Your total current expenditure for software maintenance.

   c. The source, cause, or reason for current software maintenance tasks.

   d. An evaluation of performance/reliability/quality of past software maintenance efforts.

7. Other comments:

8. Evaluation of interview questions:

Appendix 2: Description of Systems Investigated

1. Weapon System Name - Perimeter Acquisition Radar Site (PARCS)

Personnel Contacted - 1Lt Steve King HQ SAC/ADS A-271-3189

Gene Johnson Raytheon A-330-3207

System Description - The primary mission of PARCS is to provide the North American Air Defense Command (NORAD) with Early Warning Confirmation, Submarine Launched Ballistic Missile (SLBM) detection, and Attack Characterization (size of raid and impact point prediction) for Intercontinental Ballistic Missiles and SLBMs threatening the continental United States. The secondary mission is to conduct SPACETRACK operations in support of the NORAD SPADATS mission (consists of detecting, tracking, identifying, and cataloging man-made objects in space, to collect SOI and provide this information to the NORAD Intelligence Center). This station also supports tracking of new launches, identification of lost satellites, tracking of break-ups, and establishment of Tracking Impact Predictions for decaying satellites. PARCS was originally part of the Army SAFEGUARD system. The SAFEGUARD system ws declared operational in October 1975 and terminated in February 1976, due to public reaction to the cost. A subsequent decision was made to use the PAR site as a NORAD early warning sensor. The attack characterization system was operational

January 1977 under the Army, and AF personnel arrived on site in May 1977 for training. The system was transferred to the Air Force in October 1977, and a SPACETREAK capability added in December 1977.

PARCS is currently contractor maintained. Raytheon supports the system. Teledyne Brown holds the software support subcontract.

2. Weapon System Name - SLBM Phased Array Warning System - PAVE PAWS

Personnel Contacted - Capt Charles Taft HQ SAC/ADS A-271-3189

Capt Art Harriott 3900 CSS A-368-5227

System Description - The SLBM Phased Array Warning System (PAVE PAWS) consists of four long range, dual-faced phased array radars (AN/FPS-115), one one the East Coast (Otis ANGB MA), one in the Southeast CONUS (Robins AFB GA), one on the West Coast (Beale AFB CA), and one in the Southwest CONUS (Goodfellow AFB TX). Their mission is to provide warning and attack characterization of an SLBM attack against the continental United States, Alaska, and Canada. Their secondary mission is to provide warning and attack characterization of an ICBM attack against the above geographical areas. As collateral sensors, they have a tertiary mission of SPACETRACK operations providing NORAD/ADCOM Space Detection and Tracking System (SPADATS) support in the form of surveillance, tracking, reporting, and space object identification.

PAVE PAWS softtware is currently maintained by 3900 Computer Services Squadron located at Beale Air Force Base, California.

3. Weapon System Name - COBRA DANE

Personnel Contacted - Lee Severance HQ SAC/AOS
A-271-3189

Wally Moses Raytheon
1-617-358-2721 X-2983

Capt Bill Swiderek

System Description - COBRA DANE provides early warning and intelligence profiles of Soviet launches. Radar and support systems are located at Sheyma AFB, Alaska. System was developed by Raytheon Systems Development Corporation and was operational in January 1977. The software is currently maintained on-site and off-site by Raytheon.

4. Weapon System Name - Ballistic Missile Early Warning System (BMEWS)

Personnel Contacted - Major Ellis Conley HQ SAC/ADS
A-271-3189

System Description - BMEWS is a long-range high-speed radar warning system aimed at detecting enemy ballistic missile attacks from the north. The knowledge that a missile attack has been launched enables the National Command Authorities, the North American Air Defense Command (NORAD)_ and the Strategic Air Command (SAC) to take steps to counter the aggression.

Three BMEWS sites were planned and built. Site I, at Thule, Greenland, was the first to be completed in October

1960, followed by Site II at Clear, Alaska, in June 1961, and in September 1963 Site III at Fylingdales Moor in Yorkshire, the United Kingdom.

BMEWS has additionally assumed a role in satellite tracking to supplement its primary missile-warning mission. It has the capability to track all earth orbiting satellites within range of its radars. This vital data is also flashed to NORAD's Space Computational Center in Colorado Springs, where it is used to update the location of each tracked satellite.

For nearly 20 years the Aerospace Defense Command (ADCOM) was responsible for the maintenance and operation of the BMEWS. In December 1979 in a manpower savings reorganization plan, the Strategic Air Command assumed responsibility for the management of BMEWS.

The BMEWS system is currently supported by Federal Electric Services. Software maintenance functions have been frozen since 1977.

5. Weapon System - AN/FPS-85

Personnel Contacted - Capt Nick Coleman HQ SAC/ADS
A-872-1110

LTC King 20th MWS
A-883-8331

System Description - AN/FPS-85 is a phased array warning system. Their mission is to provide warning and attack characterization of SLBM attack against the United States. AN/FPS-85 also provides space craft detection and cateloging

of earth orbiting sattelites. AN/FPS-85 acts as the alternate SPADOC computational center. Software maintenance is provided by 20th MWS at Eglin AFB, Florida.

6. Weapon System - AN/FSS-7

Personnel Contacted - Capt Nick Coleman HQ SAC/ADS A-872-1110

System Description - AN/FSS-7 is a radar warning system which provides SLBM attack detection. System is located at McDill AFB. Hardware and software changes to the system are currently frozen. The system is planned for replacement in FY 85.

7. System Name - Command Center Processing and Display System - CCPDS

Personnel Contacted - LTC Kenneth P. Schaebethal HQ SAC/ADSW A-271-6363

Maj Jim Brenton HQ SAC/ADSW A-271-4505

Capt Russ Logan HQ SAC/ADSW A-271-3253

Mr. Gilbert Karr HQ SAC/ADSW A-271-6363

System Description - CCPDS is dedicated to real time receipt, processing, and common display of missile, space and atmospheric tactical warning, and attack assessment information. Common displays are defined as containing exactly the same information for all CCPDS users at the same time. CCPDS willalso support command center unique functions that do not adversely impact the common functions of the CCPDS.

These requirements are applicable to those current and future CCPDS installations at the NMCC, ANMCC, NORAD, SAC, and any other command center that may be designated in the future. Information derived from the CCPDS is required by the NCA, Joint Chiefs of Staff, CINCNORAD, and CINCSAC for their use in response decisionmaking.

Software maintenance is performed by HQ SAC/ADSW at Offutt AFB, Nebraska.

8. System Name - F-111 Digital Computer Complex - F-11D, F-111F, FB-111A Aircraft

Personnel Contacted - Lynn Bassett Sacramento ALC/MME A-633-2090

John Chow Sacramento ALC/MME A-633-2090

Bobby Ward General Dynamics 916-920-3663

Gordon Mahlman Rockwell 916-920-1215

System Description - The avionics system of the F-111 aircraft is one of the most complex and sophisticated of any in the Air Force inventory. F-111 system acquisition was the responsibility of the Air Force Systems Command (AFSC). Development and production of the aircraft were accomplished by the General Dynamics Corporation, Fort North Division, with avionics equipment provided by Rockwell International Corporation. Conractual engineering support has been procured from General Dyanmics Corporation since 1971. Rockwell International Corporation has provided similar support since 1972.

The heart of the F-111 Mark II avionics system is a Digital Computer Complex (DCC) and an Inertial Navigation System (INS), both of which employ Operational Flight Programs (OFPs). The DCC contains the General Navigation Computer (GNC) and the Weapons Delivery Computer (WDC), while the INS contains the Navigational Computer Unit (NCU). The OFPs for these computers form the brain of the F-111 Bombing and Navigation system, each performing its own function while providing a degree of redundancy in the event of failur in the other units. Each of the three digital versions of the F-111 aircraft uses a different set of OFPs which is unique to its own model. Each aircraft's OFPs undergo an eightenn month block change cycle to provide software update and changes dictated by the needs of the users and by requirements for operational enhancements and correction of deficiencies. Initiation of each aircraft's block change cycle is staggered at six months interval to efficiently utilize manpower and facilities resources.

Software maintenance is currently performed by General Dynamics and Rockwell at the F-111 Avionics Integration Support Facility (AISF) at McClellan AFB, California.

9. System Name - Programmable Indicator Data Processor (PIDP)

Personnel Contacted - Dee Ann Manning AFCCPC/ A-735-3281

System Description - PIDP is a Class V modification to fixed radar facilities using the AN/TPX-42A Air Traffic

Control Radar Beacon System. The object of the PIDP program is to replace the existing hardwired indicator data processor, and to upgrade current display capabilities of the AN/TPX-42A. The replacement is scheduled for FY 82. Air Force Communications Command is the primary user and responsible for maintaining the system software. The software is maintained at the AF Communications Computer Programming Center (CCPC) at Tinker AFB, Oklahoma.

10. System Name - Air Force Satellite Communications (AFSATCOM)

Personnel Contacted - 1Lt Tony Vander Heyden AFCCPC/

1Lt Anna Elliott AFCCPC/
A-735-5901

System Description - The Air Force Satellite Communications System will provide a satellite communication capability to satisfy high priority Air Force requirements for operational command and control of forces on a world-wide basis. Some of the communication functions satisified by AFSATCOM include Emergency Action Message (EAM) dissemination, Force Direction, Force Report Back, and CINCNET interconnectivity. AFSATCOM software is maintained by AFCCPC at Tinker AFB, Oklahoma.

11. System Name - Improved Emergency Message Automatic Transmission System (IEMATS)

Personnel Contacted - 1Lt Steve Gilbert AFCCPC/

APPENDIX 3

DATA ANALYSIS METHODS

The data which was gathered through the structured interviews was analyzed using a few fundemantal statistical approaches. The approaches are outlined below with a single problem worked for each.

Problem 1: Analysis of Percentage Data. A series of interview questions were structured to solicit a response where an entity was described by categorizing the percentage of its parts. An example of this type of question is shown below:

Section II Question 10 - Describe what percentage of time a programmer spends on the following functions:

a. Problem Analysis and Solution Design
b. Coding
c. Configuration Control Process Interace
d. Documentation Changes
e. Testing and Implementation
f. Other Functions

The objective of this type of question is to describe the current maintenance environment by establishing common relationships between the systems being investigated. This type of question was analyzed using the following approach:

Step 1: Percentage responses were first converted to rankings. This was done by assigning a ranking of 1 to the response which had the largest

percentage, a 2 to the second largest, etc. The responses to

Section II Question 10 were translated as shown in the table translation shown below:

Responses Given in Percentages

System Number	A	B	C	D	E	F
1	30	10	25	25	10	
2	30	10	25	25	10	
3	40	10	25	8	17	
4	25	15	22	23	15	
5	18	12	10	20	40	*
6	18	12	10	20	40	*
7	18	12	10	20	40	*
8	45	15	15	10	15	
9	40	15	15	10	20	
10	30	10	12	15	20	
11	32	18	5	20	25	
Mean -	29.6	12.6	15.9	17.8	22.9	
Percentage Summary						
	30	12	16	18	23	

Responses Translated to Ranking

System Number	A	B	C	D	E	F	Total
1	1	4.5	2.5	2.5	4.5		15
2	1	4.5	2.5	2.5	4.5		15
3	1	4	2	5	3		15
4	1	4.5	3	2	4.5		15
5	3	4	5	2	1		15
6	3	4	5	2	1		15
7	3	4	5	2	1		15
8	1	3	3	5	3		15
9	1	3.5	3.5	5	2		15
10	1	5	4	3	2		15
11	1	4	5	3	2		15
Sum Rj	17	45	40.5	34	28.5		165

Mean and Sum Rj values are computed as these matrixes are developed.

Step 2: The null hypothesis and alternative hypothesis of interest is set as:

H: No association between rankings

A: Association between rankings

The Kendall coefficient of concordance is used to test the null hypothesis that ranking responses are independent. The Kendall coefficient is measured in the following manner:

Kendall coefficient (w) =

$$\frac{12 \sum_{j=1}^{n} R_j^2 - 3k^2 n (n+1)^2}{n k^2 (n^2-1)}$$

where k = number of row responses

n = number of column categories

Rj = Sum of rank values per column

Thus, this problem can be solved with k = 11, n = 5 and the previously calculated Rj values.

$$w \sum_{j=1}^{n} R_j^2 = 17^2 + 45^2 + 40.5^2 + 34^2 + 28.5^2 = 5922.5$$

$$w = \frac{12(5922.5)}{5\ (121)\ (24)} = \frac{3(121)\ (5)\ (36)}{}$$

$$= .3946$$

If there is perfect agreement in ranking in all systems, then response A receives the same rank for all systems, response B receives the same rank, etc... and the resulting value of w is 1.0. If there is perfect disagreement amoung rankings, then the values of each Rj will be very close to equal and the resulting value of w will be close to 0.0. If the value of w is between 0.0 and 1.0 its significance must be further evaluated.

If the value of W is small then the ranking was independent. If W is computed to be large than the null hypothesis must be rejected in favor of the alternative hypothesis: The next step is to evaluate the resulting W value by computing the following Q statistic:

$$Q = k(n-1)W$$

and finding the associated P - value using the CHI SQUARE Distribution with n-1 degrees of freedom.

The evaluation of the Kendall coefficient in this manner is known as the Friedman Test.

Thus $Q = 11(4)\ (.3946) = 17.3624$

using CHI SQUARE with 4 degrees of freedom, $P < .01$. The data does not support the null hypothesis of independence, and there for indicates a relationship between the rankings of each system.

Step 3: If the alternative hypothesis was favored then the percentage responses were averaged; a pie chart summary of findings would display the mean responses.

Problem 2: Chi Square Contingency Table Test. A second type of problem was structured in order to identify a set of entities which were related. Section III, Questoin 3 is a sample of this type of inquiry:

Rank the top three and bottom three entities as they affect the level of software maintenance workload.

Entities:

a. Budget available

b. Size of Software (lines of code)

c. Code and documentation quality

d. Experience of maintenance Programmers

e. Level of user enhancements

f. History of error occurence

g. System Age

h. Management controls to be used

i. Maintenance-tools to be used

The matrix of responses to this question is shown below. (The top three choices are labeled 1, 2, 3; the bottom are labeled 9, 8, 7).

Rankings

System Number	A	B	C	D	E	F	G	H	I
1	3			1		9	2	8	7
2	8	9	2	3	1				7
3	8	1	3		2		9		7
4	8		2	1		3	9		7
5	3			1		9	2	8	7
6		7	2	9		3	1	8	
7	8	1			2	3		9	7
8		1		7	2	3		9	8
9	3	7	8	1		9	2		
10	8	7				9	1	2	3
11			3	2	1	9		8	7

The objective of this type of question is to identify those factors which relate/do not relate to some common entity (maintenance workload). This type of problem was anlayzed using this approach:

Step 1: Rankings responses were grouped by level as shown below:

	A	B	C	D	E	F	G	H	I
Top	3	3	5	6	5	4	5	1	1
Middle	3	4	5	3	6	2	4	4	2
Bottom	5	4	1	2	0	5	2	6	8

Step 2: The expected value for each cell of the table is 11/3. The significance of responses are then evaluated using the Chi-Square contingency table.

Total number of responses = N = 99
Number of responses in each column = C = 11
Number of responses in each row = nj = 33
Number of rows = ROW = 3
Number of columns = COL = 9

Each cell is evaluated as follows:

$$\frac{(\text{Observed Values} - \text{Expected Value})^2}{\text{Expected Value}}$$

This value for each cell is calculated and sumed.

$$T = .4 + .4 + 1.3 + 5.4 + 1.3 + .1 + 1.3 + 5.4 + 5.4 + .4 + .1 + 1.3 + .1 + 5.4 + 1.3 + .1 + .1 + 1.3 + 1.3 + .1 + 5.4 + 1.3 + 13.4 + 1.3 + 1.3 + 5.4 + 13.4$$

$T = 74.1$

A critical region of approximate size = .05 corresponds to values of T greater than 26.3 obtained from the Chi

Square distribution with (ROW-1)(CUL-1)=(3-1)(9-1)=(2)(8)=16 degrees of freedom. Since 74.1 is greater than 26.3, the null hypothesis is rejected. The conclusion is that responses are distributed differently amoung each grouping (Top, Middle, Bottom).

Step 3: Significant distribution differences are evaluated by observation. Significant factors are identified as:

C - Code and Documentation Quality
D - Experience of maintenance programmers
E - Level of user enhancements
G - System age

The degree to which each of these factors affects the level of software maintenance workload is investigated through other interview questions.

Problem 3. Range of Descriptive Values. A series of interview questions were structured to describe the current state of maintenance activities by assigning relative descriptive scoring. An example of this type of question is shown below:

Section III, Question 2. In your judgement, to what extent have the following been (or are) problems in maintaining the application system you have described?

a. Turnover of maintenance personnel -

b. Quality of application system documentation -

c. Changes made to system hardware and software -

d. - etc.

The level of the problem was rated on a 1 to 5 scale where:

5 = Major problem

4 = Somewhat major problem

3 = Minor problem

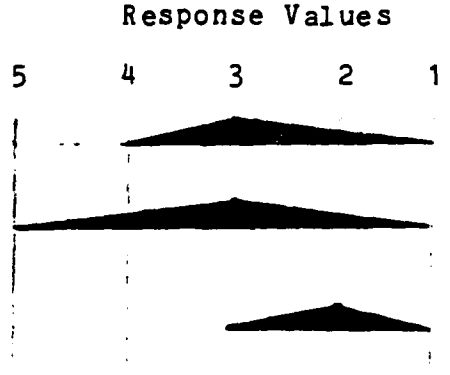
2 = Somewhat minor problem

1 = No problem

These problems were analyzed using the following approach:

Step 1: The minimum, maximum and median response values were calculated for each problem area.

Step 2: The range of values were displayed in the following manner:



a. Turnover of maintenance personnel

b. Quality of application system documentation

c. Changes made to system hardware and software

d. -

Step 3: In this case, the most significant problem areas are indicated by median values of 3 or larger. The establishment of this threshold is supported by other interview questions.

APPENDIX 4

APPLICATION OF STATISTICAL METHODS TO STRUCTURED INTERVIEW QUESTIONS

This appendix details the responses to interview questions which were statistically evaluated. The responses to other questions are summarized and discussed in Chapter 4 and Appendix 2. Each question summarized in this appendix includes an itemization of system responses, an indication of which statistical analysis method was used and intermediate and final analysis results. Thirteen systems were evaluated.

System Number	System Name
1	AFSATCOM
2	IEMATS
3	PIDP
4	FB-111A
5	F-111D
6	F-111F
7	PARCS
8	PAVE PAWS
9	COBRA DANE
10	FPS-85
11	CCPDS
12	BMEWS
13	FSS-7

System 12 and 13 have been frozen and software maintenance functions are no longer active. For this reason, many of the questions will not have a response for these two systems. Appendix 3 details the application of three major statistical methods:

Statistical Method 1: Kendalls coefficient of concordance

Statistical Method 2: Chi Square Contingency Table

Statistical Method 3: Mean/Median Summary

Question Response Summary:

1. Section II = Question 1/2

System Number	Question Response (Percentage) A	B	C	D	E
1	25	15	30	25	5
2	10	25	55	10	
3	30	15	45	5	5
4	25	30	40	5	
5	25	30	40	5	
6	25	30	40	5	
7	30	25	30	15	
8	30	25	45	10	
9	15	25	35	15	10
10	25	10	40	10	5
11	25	15	40	10	10

Statistical Method #1 Used.

Kendalls Coefficient - .4132
Q statistic = 18.1808
Chi Square test showed adoption of alternative hypothesis (p>.01).

Mean values for responses is shown below:

A - 28
B - 17
C - 39
D - 11
E - 5

2. Section II = Question 6

Maintenance Manning
(By Personnel Type)

System Number	Clerical	Programmer Analysts	Managers	Other
1	1	13	2	-
2	1	4	1	-
3	2	16	2	2
4	-	-	-	-
5	5	21	3	-
6	-	-	-	-
7	4	55	4	-
8	2	19	6	-
9	1	8	2	1
10	4	42	4	-
11	3	38	5	3

These numbers were summarized in accordance with statistical method number 3. Results are discussed in Chapter 4.

3. Section II = Question 7

Manager Tasks
(By Percentage of Time)

System Number	A	B	C	D	E
1	60	5	0	35	0
2	30	15	15	0	40
3					
4	20	60	0	20	0
5	-	-	-	-	-
6	-	-	-	-	-
7	-	-	-	-	-
8	40	20	0	10	30
9	10	30	35	20	5
10	0	65	25	10	0
11	40	20	25	15	0

Statistical Method Number 1 Used.
Kendalls Coefficient = .129
Q Statistic = 5.676
Chi Square test showed adoption of null hypothesis
(p>.1).

Mean values were not deemed meaningful.

4. Section II = Question 8

System Number	Programmer Tasks (By Percentage of Time) A	B	C	D	E	F
1	30	10	25	25	10	
2	30	10	25	25	10	
3	40	10	25	8	17	
4	18	12	10	20	40	
5	18	12	10	20	40	
6	18	12	10	20	40	
7	25	15	22	23	15	
8	45	15	15	10	15	
9	40	15	15	10	20	
10	30	10	12	15	20	
11	32	18	5	20	25	

Problem solved in problem 1 of Appendix 3.

5. Section III = Question 1A-1E

System Number	Ratings of Maintenance Factors (A-E) A	B	C	D	E
1	4	4	-	3	2
2	4.7	4	3	3	4
3	5	4	4	4	3.2
4	4	4	-	3	2
5	4	4	-	3	2
6	4	4	-	3	2
7	3.2	3	3	3	3
8	4	4	3	3	4
9	4	4	3	3	3.5
10	4	2.5	3	3	3.5
11	5	6*	3	3	4.5

Problem solved using method number 3.

Median/Mean Values are shown below:

A - 4
B - 4
C - 3
D - 3
E - 3.4

6. Section III - Question 2

Problem Area	System Number 1	2	3	4	5	6	7	8	9	10	11
a.	4	3	3	1	1	1	4	3	3	3	4
b.	4	1	1	2	2	2	4	3	3	5	2
c.	1	1	2	1	1	1	1	2	2	3	2
d.	1	5	2	2	2	2	4	3	2	2	3
e.	4	1	3	1	1	1	-	1	2	3	2
f.	5	1	5	3	3	3	4	2	3	3	2
g.	4	3	2	4	4	4	-	2	3	1	3
h.	4	3	3	1	1	1	2	1	2	-	-
i.	1	1	1	1	1	1	-	1	1	1	3
j.	5	1	3	1	1	1	1	1	1	3	2
k.	1	3	4	2	2	2	3	2	2	3	2
l.	4	1	2	4	4	4	5	2	2	1	2
m.	3	2	4	2	2	2	1	1	2	1	3
n.	2	1	1	3	3	3	-	3	2	-	3
o.	3	2	3	2	2	2	1	1	1	2	4
p.	1	1	3	2	2	2	-	2	2	2	1
q.	5	1	3	2	2	2	1	2	1	2	3
r.	1	1	3	2	2	2	1	2	2	1	2
s.	1	3	2	2	2	2	1	3	1	1	2
t.	1	1	2	2	2	2	1	3	3	1	3
u.	1	4	3	2	2	2	2	1	2	1	2
v.	4	1	2	4	4	4	2	1	3	1	2
w.	1	1	-	4	4	4	5	2	1	2	4
x.	1	5	-	3	3	3	1	3	1	1	4

Response values are summarized in Table 4 of Chapter 4.

7. Section IV - Question 6

System Number	Responses (Y/N) A	B	C	D
1	N	N	Y	Y
2	N	Y	N	Y
3	N	Y	Y	Y
4	N	Y	N	Y
5	Y	Y	N	N
6	N	Y	N	Y
7	N	N	N	Y
8	Y	Y	Y	Y
9	N	N	Y	N
10	N	N	Y	Y
11	N	Y	N	Y

This question was evaluated using method number 1. There ws no correlation of responses between systems.

Kendalls coefficient - .103
Q statistic - 4.532.

The null hypothesis of independent responses was accepted with ($p>.1$).

Summary data was not calculated.

Vita

James Patrick Joyce was born 29 May 1955 at Colorado Springs, Colorado. He received the degree of Bachelor of Science in Computer Science from Oregon State University in June 1977. Upon graduation, he received his commission through Air Force ROTC. From October 1977 to May 1981, he was assigned to the Tactical Fighter Weapon Center (TFWC) at Nellis AFB, Nevada. During this assignment, Capt Joyce served as Computer Systems Support Officer and as Chief of the Data Processing Plans and Requirements Branch. In June 1981, he entered the Air Force Institute of Technology. Capt Joyce will receive the degree of Masters of Science in Information Systems from AFIT in December 1982.

Permanent Address: 1305 Stella Drive
Colorado Springs Colorado
80908

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GCS/MA/82D-5	2. GOVT ACCESSION NO.: AD-A124 458
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): A Study of the Software Maintenance Process of Air Force Weapon Systems	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): James P. Joyce, Capt USAF	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Test and Evaluation Center (AFTEC) Kirtland AFB, New Mexico	12. REPORT DATE: Dec 1982
	13. NUMBER OF PAGES: 162
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Aproved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release: IAW AFR 190-17

LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AIC)
Wright-Patterson AFB OH 45433

Approved for public release; IAW AFR 190-17

FREDERIC C. LYNCH, Major, USAF
Director of Public Affairs

4 JAN 1983

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SOFTWARE MAINTENANCE
AIR FORCE WEAPON SYSTEM SOFTWARE
SOFTWARE MAINTAINABILITY
SOFTWARE MANAGEMENT
SOFTWARE ACQUISITION
SOFTWARE SUPPORT

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The increasing cost of software maintenance is becoming a critical issue. This master's thesis profiles Air Force software maintenance activities and provides recommendations for improving management and programming efforts. The software maintenance

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

activities of thirteen Air Force weapon systems were investigated through structured interviews of key management and programmer personnel. Analysis of interview responses resulted in the identification of four factors which effect the level of maintenance effort: system age, personnel experience, documentation and code quality, and level of user enhancements. Interview responses also identified three major management policy issues which cause the Air Force to make excess resource commitments to software maintenance functions:

1) Personnel experience;
2) Software development and life cycle planning, and
3) Software development and maintenance standards.

Management policy recommendations were developed to reduce the impact of these conditions on Air Force weapon system software maintenance activities.